AFRL-RI-RS-TR-2011-041

INVESTIGATION OF ZERO KNOWLEDGE PROOF APPROACHES BASED ON GRAPH THEORY

FEBRUARY 2011

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2011-041 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/

MICHAEL S. GUDAITIS
Chief, Platform Connectivity Branch

/s/

WARREN H. DEBANY JR, Technical Advisor
Information Grid Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
February 2011	Final Technical Report	May 2009 – July 2010

4. TITLE AND SUBTITLE

INVESTIGATION OF ZERO KNOWLEDGE PROOF APPROACHES BASED ON GRAPH THEORY

5a. CONTRACT NUMBER
In House

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Victoria Horan
Michael Gudaitis

5d. PROJECT NUMBER
RIGD

5e. TASK NUMBER
IH

5f. WORK UNIT NUMBER
CM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/RIGD
525 Brooks Road
Rome, NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site
26 Electronic Parkway
Rome NY 13441

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2011-041

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2010-6755.
Date Cleared: 3 January 2011.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Zero-knowledge proof systems have many characteristics that are desirable for determining trustworthy parties in an airborne networking environment. One approach is to base zero-knowledge proof systems on the instances and solutions of NP-complete problem. This report investigates this approach with a focus on the graph theory problems within the NP-complete and NP-hard classes.

15. SUBJECT TERMS

Airborne Network Protocol, Zero Knowledge Proof, Graph Isomorphism

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			MICHAEL S. GUDAITIS
U	U	U	UU	107	19b. TELEPHONE NUMBER (*Include area code)* N/A

Approved for Public Release; Distribution Unlimited.

TABLE OF CONTENTS

LIST OF FIGURES

1. EXECUTIVE SUMMARY

Airborne mobile ad hoc network (MANET) environments require a quick and lightweight method for authentication. These constantly changing airborne networks (AN) need some way to identify trustworthy users without a third-party involved. One possible answer to these requirements is the zero-knowledge proof method. Zero-knowledge proof systems provide an interactive approach for an entity to prove the possession of private knowledge without revealing any information about it. Successful challenge/response interactions between a Prover and Verifier provide a confidence level of trust to the Verifier that the Prover indeed possesses the private information. The fact that the private knowledge is never revealed provides benefits towards achieving a protocol that is secure against eavesdroppers. The desirable characteristics in a zero-knowledge proof system for airborne MANETs are (1) low amount of information (i.e. bits per transaction) transferred between parties, (2) low number of iterations of the protocol needed to establish trust, and (3) low probability that an untrustworthy party is able to establish trust. Characteristics (1) and (2) provide a lightweight protocol, while characteristic (3) ensures that the protocol is strong.

Since zero-knowledge proof systems require a verifier to check that the information received from the prover exhibits knowledge of the private input, the base problem (for which the private input is the solution) must be easily verifiable. However, for the protocol to be hard to cheat, we must have a base problem that is difficult to solve from scratch. This leads us to consider base problems that fall in the class of NP-complete problems – computationally expensive decision problems in which a positive solution can be checked in polynomial-time. This report investigates the graph theory subset of the class of NP-complete problems and their use as base problems for zero-knowledge proof systems. In particular, the problems examined most in-depth are all related to either the sub-graph isomorphism problem or the graph coloring problem.

In this paper, several approaches are formulated into a zero-knowledge proof system, and their characteristics are examined. Examples of the following graph problems are given: sub-graph isomorphism graph isomorphism, independent set, longest path problem, Hamiltonian cycle problem, graph 3-coloring, equitable 3-coloring, satisfiability, and graph partitioning.

Considering the problem classes discussed, the least promising problem is the satisfiability problem, due to very efficient algorithms that are able to solve enormous problem instances very quickly. The most promising group appears to be the sub-graph isomorphism class. The zero-knowledge proof systems associated with this class of problems are relatively lightweight in comparison with the other problem classes, and several of the problems in the class have many difficult instances and few efficient algorithms. The protocols have average strength in terms of the proof systems for graph-based problems. In comparison, the graph coloring class has many difficult instances for the problems, but the existing zero-knowledge proof systems are relatively easy to cheat. This then implies that the proof systems are not as strong as those in the sub-graph isomorphism class.

2. INTRODUCTION

In an airborne networking (AN) environment, the mobility of the network users necessitates an agile authentication system. Zero-knowledge proof systems allow an interaction between parties to determine trustworthiness in a quick and effective manner. In order to make these interactions as fast and secure as possible, they are most often based on problems from the NP-complete class, which contains many graph theory problems. A strong and lightweight zero-knowledge protocol must satisfy the following criterion: it must have a small number of bits transferred between parties, it must require few iterations to achieve a given trust level, and it must be difficult for a cheater to pass as trustworthy.

This report is outlined as follows. Section 2 continues to provide the necessary background information in graph theory, complexity theory, and zero-knowledge proof systems. Sections 3 through 5 discuss individual problems that zero-knowledge proof systems can be based on. Section 6 presents our conclusions and future work. Section 7 lists the relevant references, and the appendix expands upon that list to provide an annotated bibliography.

2.1 Graph Theory Background

This section is meant as a guide to some of the graph theoretic terms and concepts employed in this report. For a more extensive reference, it is recommended that the reader consult a textbook such as Diestel's *Graph Theory* (Diestel 2006).

A *graph* is a pair $G = (V, E)$ such that E is a subset of $V \times V$, where V is the set of *vertices* and E is the set of *edges* in the graph. Vertices can also be called *nodes*. An edge is *incident* to a vertex if the vertex is one of the edge's endpoints. Two vertices are *adjacent* (also called *neighbors*) if they are connected by an edge. The *degree* of a vertex (or *valency*) is the number of edges incident to it. An *adjacency matrix* representation of a graph is a matrix in which the rows and columns represent the vertices and an entry equal to 1 in row u and column v implies the existence of an edge between vertices u and v, while an entry equal to 0 implies that there is no edge between u and v.

In discussing graphs, we use the following terms. A *simple graph* is a graph in which there is at most one edge between distinct vertices and there are no edges from a vertex to itself (called a *loop*). We will only deal with simple graphs in this report. The *complement* $\overline{G} = (\overline{V}, \overline{E})$ of a graph $G = (V, E)$ is the graph in which $\overline{V} = V$ and (x, y) is an edge in $\overline{G}$ if and only if it is a non-edge in G, i.e. $\overline{E} = \{(x, y) \notin E : x, y \in V\}$. A *regular graph* is a graph in which every vertex has the same degree. A *labeled graph* is a graph with labels (distinct or not distinct) placed on the vertices or the edges. A *weighted graph* has edge labels that denote a *weight* on an edge. These weights could represent distance or some other measure. An *unweighted graph* has no labels on the edges. This could also be defined as a graph with all edge weights equal to 1. The *empty graph* is the graph $G = (V, E)$ with $E = \emptyset$. A *complete graph* is a graph in which

every edge possible is present, i.e. for every pair of distinct vertices $x, y \in V$, there is an edge $(x, y) \in E$.

There are many terms for describing the structures present within a graph. A *sub-graph* $G' = (V', E')$ of a graph $G = (V, E)$ is a graph in which $V' \subseteq V$ and $E' \subseteq E$. We denote that G' is a sub-graph of G by writing $G' \subseteq G$. An *induced* sub-graph $G' = (V', E')$ of a graph $G = (V, E)$ is a sub-graph of G in which $E' = \{(x, y) \in E : x, y \in V'\}$. To indicate an induced sub-graph, we write $G' = G[V']$. A *path* is a sequence of vertices and edges such that no vertices and no edges are repeated. A *cycle* is a path with the exception that the first and last vertices are the same. A *Hamiltonian cycle* or *path* is a cycle or path that travels through every vertex in the graph. The *length* of a path or cycle is the number of edges. An *independent set* is a set $S \subseteq V$ in a graph $G = (V, E)$ such that the edge set of $G[S]$ is an empty set. A *clique* is the complement of an independent set, i.e. a sub-graph $G' = (V', E')$ of $G = (V, E)$ such that $E' = \{(x, y) : x, y \in V'\}$.

Much of this report utilizes the concept of a graph isomorphism. Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there exists a bijective function $f : V(G) \to V(G')$ such that $(x, y) \in E$ if and only if $(f(x), f(y)) \in E'$. Less formally, two graphs are isomorphic if they exhibit the same structures. A *sub-graph isomorphism* is an isomorphism from a graph G to a sub-graph $H \subseteq G'$.

2.2 NP-Completeness

This section is an introduction to some of the theory of NP-completeness and complexity theory that is utilized in this report. For a more extensive reference, it is recommended that the reader consult a textbook such as Skiena's *Algorithm Design Manual* (Skiena 2008).

The complexity classes involved in this report are primarily the classes P (Polynomial-time) and NP (Non-deterministic Polynomial). Because we will not discuss Turing machines in this report, we will state the somewhat less formal definitions of the complexity classes. The class NP is the class of decision problems for which any yes-instance has a solution that is verifiable in polynomial time. The class P contains all decision problems that can be solved in polynomial time, and hence also have solutions that can be verified in polynomial time, implying that P ⊆ NP.

A problem *L* in the class NP is in the subclass of NP-complete problems if every problem in NP can be reduced to the problem *L* in polynomial time. A reduction from problem *K* to problem *L* is an algorithm which takes as input an arbitrary instance of problem *K* and outputs an instance of problem *L*. Given this definition, it is clear that the class of NP-complete problems contains the hardest problems in the class NP, as an easy solution for one NP-complete problem leads to an easy solution for all problems in the class NP.

Because of the work published by Cook (Cook 1971), which proves that satisfiability is the first NP-complete problem, proving an NP-complete problem is somewhat easier than the definition implies. As Cook proved that satisfiability is reducible to any problem in NP, in order

to prove a problem is in the class of NP-complete problems, we need only prove that a known NP-complete problem L reduces to our problem. Then through Cook's theorem and following the chain of reductions from satisfiability to L, we have shown that every problem in the class NP reduces to our problem.

2.3 Zero-Knowledge Proof Systems

This section is an introductory guide to some of the theory and concepts of zero-knowledge proof systems that are used in this report. For a more extensive reference, it is recommended that the reader consult a textbook such as Simmons's *Contemporary Cryptology* (Simmons 1992).

We begin with the notion of an interactive proof system. An *interactive proof system* is an interaction between two participants, called the *prover* and the *verifier*, in which the prover attempts to prove some fact (or knowledge of some private input) to the verifier. An interactive proof system is formally defined as a protocol based on a decision problem which satisfies the following properties:

Completeness: Each yes-instance of the decision problem leads to acceptance by the verifier with probability at least $1 - n^{-k}$ for any constant $k > 0$, where n is the size of the problem instance.

Soundness: Each no-instance of the decision problem leads to rejection by the verifier with probability at least $1 - n^{-k}$ for any prover (honest or cheating).

A *zero-knowledge proof system* is an interactive proof system with an additional requirement: the *zero-knowledge property* must be satisfied. The zero-knowledge property ensures that the verifier cannot gain any information from the interaction with the prover that could not have been determined alone. This also guarantees that any eavesdropper cannot gain knowledge by listening to the conversation. In a zero-knowledge proof system, the interaction begins with the prover presenting a commitment to some information about the graph, which is followed by a challenge from the verifier. The challenge consists of a request for specific information about the problem instance. To prove that the zero-knowledge property is satisfied in such an interaction, we use simulators. This proof method of the zero-knowledge property is structured as follows:

1. Verifier simulates the prover
    - Given the set of possible challenges, the verifier randomly and uniformly decides which challenge to commit to a correct response to.

2. Verifier simulates the verifier
    - Using a probabilistic algorithm, the verifier decides which challenge to send.

- If the challenge does not match what was committed to in step 1, the verifier backs up the algorithm to the state it was in at the beginning of step 1 and starts the simulation over.

If the process above generates a conversation that is the same as one that could have been generated with an honest prover (one that is different from the verifier), then the zero-knowledge property is satisfied.

An easily understood example of a zero-knowledge proof system illustrates that the prover knows how to solve a Rubik's cube (the algorithm is the private input). The verifier scrambles a Rubik's cube and hands it to the prover. The prover turns away so that the verifier is unable to see the cube, and then attempts to solve the puzzle. If the prover knows the algorithm, this will be an easy task and the prover will quickly hand a solved Rubik's cube back to the verifier. If the prover does not know the algorithm, the prover may or may not be able to solve the puzzle. Most likely, a prover that does not know how to solve a Rubik's cube will not easily solve the puzzle quickly several times in a row, and so the verifier will (eventually) see that the prover does not know the algorithm. A prover that does know the algorithm will quickly solve the cube as many times as the verifier wishes.

An important component of a zero-knowledge proof system is the *commitment*. Zero-knowledge proof systems usually require that the prover has some method of "locking up" information about the problem instance prior to receiving the verifier's challenge. Otherwise, the prover would be able to manufacture a response to the verifier's challenge, and this new response may or may not be consistent with the problem instance. For example, in the Rubik's cube zero-knowledge proof system discussed in the last paragraph, the verifier may require that the prover remain in the same room so that the verifier can be sure that the prover returns the same Rubik's cube given out in the beginning. The structure of a zero-knowledge proof system is such that a response to any one challenge does not reveal the private input, but responding to all challenges reveals the private input. Hence the prover must be able to simultaneously commit to all challenge responses, but reveal each one individually.

There are several different methods for the prover to commit by. The simplest to understand (but least practical to implement) is to use locked boxes. The prover breaks up the graph into pieces, which are stored in locked boxes. Then the prover determines which boxes to open by observing the verifier's challenge. Note that the problem type will determine how the graph is broken up and stored. Another method of commitment is through encryption by keys. If the prover is able to generate keys, then the prover may encrypt the graph and send the encrypted copy to the verifier. Then depending on the verifier's challenge, the prover sends keys for decrypting the information necessary to answer the challenge. The methods used vary and is usually determined by the implementation of the proof system.

3. METHODS, ASSUMPTIONS AND PROCEDURES

3.1 SUB-GRAPH ISOMORPHISM CLASS

The sub-graph isomorphism class contains many NP-complete problems that can be obtained by a reduction from the sub-graph isomorphism problem. Figure 1 illustrates the structure of the sub-graph isomorphism problem and its related subproblems. Note that the chart lists the most general problem at the top (sub-graph isomorphism) and each subproblem allows more problem restrictions than the superproblem does. For example, graph isomorphism is a more specific instance of sub-graph isomorphism in that it requires that $H = G_2$. Thus if a problem is NP-

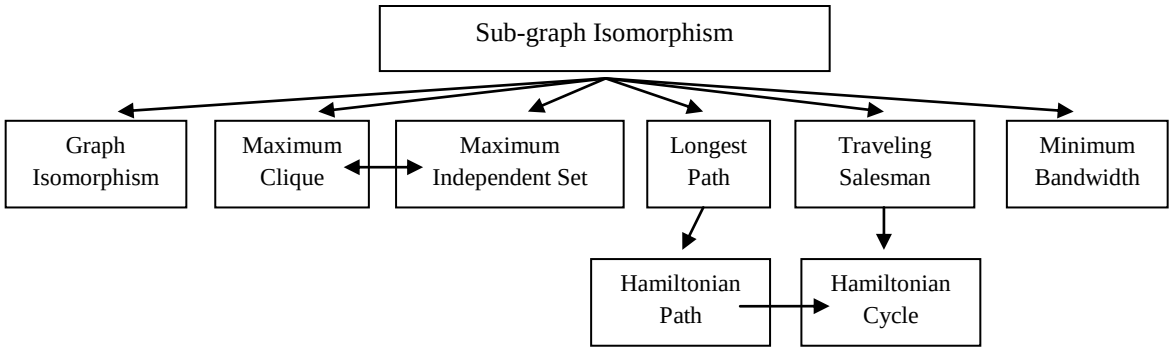


Figure 1: The subproblem structure of the sub-graph isomorphism problem

complete, all problems that contain it as a subproblem must also be NP-complete, but not vice versa.

3.1.1 Sub-graph Isomorphism Problem

The general sub-graph isomorphism problem is stated as follows: Given two graphs G_1 and G_2, is there a sub-graph H of G_2 such that G_1 is isomorphic to H? The sub-graph isomorphism problem (SGI) is an NP-complete problem (Garey and Johnson 1979), and has many well-known subproblems associated with it.

3.1.1.1 Algorithms

In the world of NP-complete problems, there are two ways to define what makes a "good" algorithm. The first is a theoretical definition, in which the computational complexity of the problem is reduced for either the general class of all instances or for a specific class of subproblems. The second is an experimental definition. Using different methods of attacking the problem, we try to create an algorithm that will solve most instances of the problem in a short amount of time, but may not be very efficient in some rarely-occurring worst-case instance.

With many useful applications, there is a significant amount of research being done on solving the sub-graph isomorphism problem using both approaches.

Theoretical Results:

Algorithms classified as theoretical results are aimed at lowering the computational complexity bounds that currently exist for solving the sub-graph isomorphism problem. These algorithms generally are not implemented or tested against each other on actual instances of the problem. Since the general sub-graph isomorphism problem is known to be NP-complete, these algorithms tend to restrict the problem in some manner in order to make the problem easier to solve.

Many algorithms exist for solving the sub-graph isomorphism problem on specific classes of graphs. For example, when considering the class of planar graphs we can reduce the running time significantly. Using a dynamic programming method, Dorn has developed an algorithm with running time on the order of $2^{\mathcal{O}(k)}n$ where the graph *H* has *k* nodes and *G* has *n* nodes (Dorn 2009). This implies that if we consider restricting the problem so that the number of nodes in the sub-graph *H* is fixed, then $2^{\mathcal{O}(k)}$ becomes a constant and hence the algorithm is linear.

Another restriction on the set of graphs that has seen good results is to consider only graphs of bounded tree-width. To define tree-width, we need a few other definitions first (Alon, Yuster and Zwick 1995).

Definition 1: Let $G = (V, E)$ be a graph. A *tree-decomposition* of *G* is a pair

$$(\{x_i : i \in I\}, T = (I, F))$$

where *T* is a tree and $\{x_i : i \in I\}$ is a collection of subsets of *V* such that:

1. $\bigcup x_i = V$,
2. $\forall (v, w) \in E, \exists i \in I$ such that $\{v, w\} \subseteq x_i$, and
3. $\forall v \in V$, when we restrict *T* to the vertex set $\{i \in I : v \in x_i\}$, we still have a connected tree.

Definition 2: Let *G = (V, E)* be a graph. Let the set of all tree-decompositions of *G* be denoted *TD(G)*. The *tree-width* of *G* is:

$$TW(G) = \min_{T \in TD(G)} \max_{i \in I} |x_i| - 1.$$

One of the most recently published randomized algorithms for solving the sub-graph isomorphism problem is shown to have running time $\mathcal{O}^*(2^k n^{2t})$ when the tree-width of *H* is at most *t* (Fomin, et al. 2009)[1].

[1] "$\mathcal{O}^*(\,)$ notation hides factors polynomial in the instance size *n* and the parameter *k*" - (Fomin, et al. 2009)

While theoretical results are useful in determining what is possible and impossible in terms of creating new algorithms for solving the sub-graph isomorphism problem, these methods are not always practical or useful for implementation and applications. The algorithm may appear fast in terms of Bachmann-Landau notation (Big-Oh notation), but this can often be misleading. For example, if an algorithm has computational complexity $\mathcal{O}(n)$, it is possible that the exact running time has some enormous constant term, say $10^{10^{10^{\cdots}}}$. In this case, even though the algorithm is linear, a running time of $10^{10^{10^{\cdots}}}n$ is going to be very costly. For solving specific instances of the problem, it may be more efficient to consider an algorithm that has worse computational complexity in a worst case scenario, but good experimental results on large databases of graphs.

Experimental Results:

In terms of algorithms that are practical to use, it is necessary to review the results obtained by implementing the algorithm and testing it on several databases of graphs. While it is not possible to test the algorithm on every possible graph, we can often get a good idea of how useful an algorithm will be by running it on specific classes of graphs and instances that are known to be difficult. The algorithms discussed in this section currently appear to be the most popular for comparing new algorithms against, and so are generally understood to be the fastest algorithms currently available. Also included are several new algorithms that appear to perform quite well against the existing front-runners.

VF2 and Ullman's Algorithm are the most popular choices for efficient sub-graph isomorphism solvers. Published this year (2010) are two different filtering algorithms that seem promising. Filtering algorithms aim to reduce the number of possible target vertices in the larger graph for each vertex in the smaller graph to be mapped to under an isomorphism. By repeatedly reducing the set of target vertices, the filtering aims to eventually obtain a target set of size one, in which case the mapping is clear (Solnon 2010).

Ullman's algorithm, published in 1976, is surprisingly still a popular and fast sub-graph isomorphism solver. This algorithm uses a backtracking method to solve the problem in an efficient manner, in most cases (Ullmann 1976). However, it is often costly and outperformed by newer methods when it comes to larger instances.

Arguably the best solver available, VF2 (also referred to as VFLib) is an algorithm for solving both the graph and sub-graph isomorphism problems. By defining certain feasibility rules, VF2 is able to reduce the number of possible options and repeatedly extend a partial matching until the correct sub-graph is found (Cordella, et al. 2004).

The filtering method ILF (Iterative Labeling Filtering) begins with an initial labeling of the vertices in both graphs by some invariant property such as vertex degree. An invariant property of a graph is one that remains constant under isomorphisms. From this it is

immediately clear that two vertices with different labels cannot possibly be mapped to one another. The algorithm then expands the lists as multisets (sets with repetition allowed) by adding the labels of adjacent nodes. This process is repeated as many times as desired. At each step, the algorithm uses an auxiliary bipartite graph to determine the compatibility of two vertices (Zampelli, Deville and Solnon 2010 (to appear)).

A new algorithm, AllDifferent-Based Filtering, introduced by Christine Solnon (Solnon 2010) uses a method known as "local all different" (LAD). For each vertex *u* in our sub-graph *H*, and for each possible target vertex *v* in *G* for *u*, the algorithm constructs a bipartite graph with vertex set (N(*u*), N(*v*)), where N(*u*) is the set of all vertices that are adjacent to *u* in *H*. The edges in this bipartite graph are of the form (*u'*, *v'*), where *v'* is a possible target vertex for *u'*. Then the algorithm searches for a matching in this bipartite graph (an independent set of edges) that covers all of N(*u*). If no such matching exists, *v* is no longer considered as a possible target for *u*.

An alternative approach to the problem is to formulate it in a way that takes advantage of efficient solvers from other problems and areas. One such approach formulates the sub-graph isomorphism problem as an integer linear program (LeBodic, et al. 2009). We define the variables of this linear program as follows. We define G = (N, L) and H = (V, E).

- For all pairs of vertices $(i,k) \in V \times N$, define $x_{i,k} = \begin{cases} 1, & if \quad i \mapsto k \\ 0, & otherwise \end{cases}$
- For all pairs of edges $(ij,kl) \in E \times L$, define $y_{ij,kl} = \begin{cases} 1, & if \quad ij \mapsto kl \\ 0, & otherwise \end{cases}$

We define the constraints as follows:

- Every vertex of H maps to a unique vertex of G

$$\sum_{k \in N} x_{i,k} = 1, \quad \forall i \in V \tag{1}$$

- Every edge of H maps to a unique edge of G

$$\sum_{kl \in L} y_{ij,kl} = 1, \quad \forall ij \in E \tag{2}$$

- Every vertex of G is targeted by at most one vertex of H

$$\sum_{i \in V} x_{i,k} \leq 1, \quad \forall k \in N \tag{3}$$

- If $i \mapsto k$ then any edge starting in i maps to an edge starting with k

$$\sum_{kl \in L} y_{ij,kl} = x_{i,k}, \quad \forall k \in N, \forall ij \in E \tag{4}$$

- If $j \mapsto l$ then any edge ending in j maps to an edge ending in l

$$\sum_{kl \in L} y_{ij,kl} = x_{j,l}, \ \forall l \in N, \forall ij \in E \quad (5)$$

Combining these, it is clear that if we find a solution to this set of constraints then we have a graph-sub-graph isomorphism. Hence, the objective function of this integer linear program is irrelevant. If there is some additional information (for example, if the problem is geometric), then there may be a useful objective function (such as minimizing the distances between nodes). However if not, we can set the objective function to be some irrelevant constant function. This gives us the general linear program:

Max	$\sum_{i \in V} 0$	
Subject to	$\sum_{k \in N} x_{i,k} = 1,$	$\forall i \in V$
	$\sum_{kl \in L} y_{ij,kl} = 1,$	$\forall ij \in E$
	$\sum_{i \in V} x_{i,k} \leq 1,$	$\forall k \in N$
	$\sum_{kl \in L} y_{ij,kl} = x_{i,k},$	$\forall k \in N, \forall ij \in E$
	$\sum_{kl \in L} y_{ij,kl} = x_{j,l},$	$\forall l \in N, \forall ij \in E$
and	$x_{i,k}, y_{ij,kl} \in \{0,1\},$	$\forall (i,k) \in V \times N$
		$\forall (ij,kl) \in E \times L$

Figure 2: Integer Linear Program for SGI

Formulating the linear program is now done, and all that is needed is an efficient solver to work on it. The testing done in (LeBodic, et al. 2009) is on very application-specific instances (architectural floor plans) making it difficult to evaluate in terms of the more general instances seen in the graph databases.

Testing of Experimental Algorithms:

While most newly published algorithms are compared against either or both of Ullman's Algorithm and VF2, most are not compared to any other recently developed methods. One difficulty in comparing the algorithms is that only recently has it become popular to use publicly available graph databases for testing in a consistent manner, making it difficult to evaluate and interpret the results that are reported.

The two main databases that are available for testing sub-graph isomorphism algorithms are the GraphBase database and the VFLib database. Several authors also have created their own

classes of graphs for testing, such as the scale-free networks database created in (Zampelli, Deville and Solnon 2010 (to appear)). The Stanford GraphBase[2] database provides generators to create various different classes of graphs and is available free (Knuth 1993). The VFLib Graph Matching Library was created specifically for the graph isomorphism and sub-graph isomorphism problems and is also available free of charge (P. Foggia 2001).

On most large instances of the sub-graph isomorphism problem, VF2 outperforms Ullman's Algorithm (see, for example, Figure 15 in (Lipets, Vanetik and Gudes 2009). For this reason, many authors compare their algorithm only with VF2. One analysis and comparison of VF2, ILF, and LAD, can be found in (Solnon 2010). This paper shows that LAD usually outperforms both VF2 and ILF when run on both the GraphBase database and the VFLib database. However, one thing that is clear is that no solver performs best in every case.

3.1.1.2 Existing Zero-Knowledge Proofs

The zero-knowledge proof systems for the sub-graph isomorphism problem take as public input two graphs, G_1 and G_2, and as private input a sub-graph isomorphism $\phi: G_1 \rightarrow H$, where H is a sub-graph of G_2. The simplest zero-knowledge proof system for sub-graph isomorphism, ZKP1, is illustrated in Figure 3. The prover (P) randomly permutes the graph G_2 to obtain an isomorphic graph $\pi(G_2) = G'_2$ and then sends G'_2 to the verifier (V). V then chooses a random bit c, which is sent to P. If $c = 0$, P sends π to V and V checks that G'_2 was formed correctly from G_2 and π. If $c = 1$, then P sends $\pi\phi$ to V and V checks that $\pi(\phi(G_1))$ is a sub-graph of G'_2 and is isomorphic to G_1. Depending on the P's response to the challenge, V will decide whether to accept P and continue in another iteration of the protocol or to reject P and stop communication.

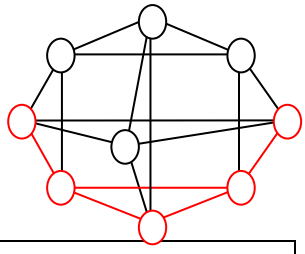
Both isomorphisms (ϕ and π) are private in the beginning, but if the verifier chooses $c = 0$ the entire isomorphism π is revealed, whereas if the verifier chooses $c = 1$ only parts of the isomorphism π are revealed. In neither case is any of the isomorphism ϕ revealed to the verifier. If $c = 0$, the verifier learns the isomorphism π, but is unaware of any information regarding the sub-graph's location. If $c = 1$, the verifier learns information about the structure of the sub-graph, but does not know the isomorphism π and hence knows nothing about the location of the sub-graph.

It is also possible to construct a similar zero-knowledge proof system that involves hiding the permuted graph in a larger graph (Grigoriev and Shpilrain 2008). However, in this modification of the protocol, either choice that the verifier can make for c requires that the prover send G'_2 to the verifier, which leaves the larger graph as an unnecessary addition to the protocol.

[2] Available at http://www-cs-faculty.stanford.edu/~knuth/sgb.html

Common Input: The bull graph (G_1) and the Payley graph (G_2) of order 9 (shown)

Private Input: An isomorphism ϕ from H to a sub-graph of G (H is shown in red)



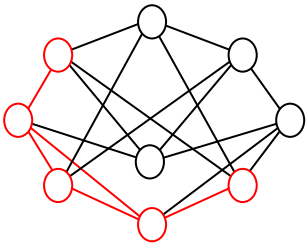
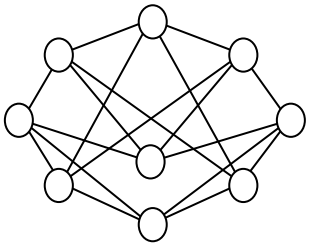
Prover		Verifier
1. Chooses an isomorphism π: $G_2 \rightarrow G'_2$. 2. Sends G'_2 to the verifier.		 3. Chooses a random bit c. 4. Sends c to the prover.
$c = 0$	5. Sends π to the verifier.	6. Checks that $\pi(G_2) = G'_2$ and that π is a valid isomorphism.
$c = 1$	5. Sends $\pi(\phi(H))$ to the verifier.	6. Checks that $\pi(\phi(H))$ is a sub-graph of G'_2 that is isomorphic to G_1.

Figure 3: ZKP1 for the sub-graph isomorphism problem example

3.1.1.3 Discussion of Existing Protocols

Considering the algorithms that exist for the graph isomorphism problem (Nauty, VF2, etc.), the protocol ZKP1 is not very secure. For example, the verifier could use an effective graph isomorphism algorithm after receiving the graph G'_2 in step 2. This enables the verifier to uncover the isomorphism, π^{-1}: $H \rightarrow G_1$. If the verifier chooses to send $c = 1$ to the prover, then $\pi(\phi(G_1))$ is revealed by the prover, and so the verifier has available both π^{-1} and $\pi(\phi(G_1))$. This would allow the verifier to discover $\phi(G_1)$, and using the graph isomorphism algorithm again would determine ϕ, the prover's private input.

3.1.1.4 Establishing a Better Protocol

A slight modification to ZKP1 to establish a more secure protocol involves committing to the permuted graph that is transferred in step 2. This alteration works fine until the last step that

occurs, in the case that the verifier chooses $c = 1$. If the verifier chooses $c = 1$, then the prover must reveal where the isomorphic sub-graph $\pi(\phi(G_1))$ is located in $\pi(G_2)$. In order for the verifier to check that $\pi(\phi(G_1))$ is in fact a sub-graph that is isomorphic to G_1, the verifier must be able to solve this particular instance of the graph isomorphism problem very quickly. Thus for this change in protocol to be effective, we must be sure that the smaller graph involved is one in which some graph isomorphism algorithm works well, and yet the problem instance as a whole must be difficult for all sub-graph isomorphism algorithms, which is not an easy task.

Another small adjustment to fix the faults described is illustrated in Figure 4, referred to as ZKP2. In this protocol, the prover sends the permutation $\phi\pi$ as well as decommitment information to reveal the edges of the sub-graph $\pi(\phi(G_1))$. The verifier would then be able to check that $\pi(\phi(G_1))$ is isomorphic to G_1. Fortunately, even with increasing the amount of information transferred, the zero-knowledge property is still satisfied. Since the verifier does not know π or ϕ individually, the verifier is unable to determine π or ϕ alone from the composition $\phi\pi$. Also, since the verifier is only shown the entries of the permuted adjacency matrix that correspond to edges of the sub-graph, the verifier cannot uncover the initial permutation ϕ unless the verifier is able to solve the sub-graph isomorphism problem.

The protocol ZKP2 is a valid zero-knowledge proof system for the sub-graph isomorphism problem. (Note: All proofs of zero-knowledge protocols in this report are based on the proof style of Blum (Blum 1986).)

Claim: ZKP2 is a zero-knowledge proof system for the sub-graph isomorphism problem.
Proof:

Completeness: If the prover has a yes-instance x of SGI, then the verifier will accept x with probability 1.

Soundness: If the prover has a no-instance y of SGI, the prover will be caught only when the verifier chooses $c = 1$. Since c is chosen uniformly and randomly by the verifier, the probability that the verifier will reject y is $1/2$ in each round. This implies that the probability that the verifier does not reject y after c rounds is at most $\left(\frac{1}{2}\right)^c = 2^{-c}$.

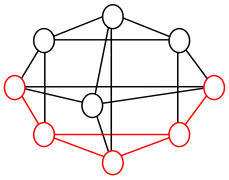
Zero-Knowledge Property: Suppose the verifier is attempting to extract useful information from his conversation with the prover. Then the verifier can, in the same manner, extract the same information even without the aid of the prover. In each round he does the following:

Begin.

Verifier simulates the *prover*. The verifier flips a fair coin and, according to the outcome of the coin, commits to either the graph G_2 or a copy of G_1 embedded into an arbitrary n-vertex graph. G_2 is committed to in the same way the prover would have done so. The sub-graph is committed to in the way the prover would have committed to such an isomorphic sub-graph in G_2. Then, acting as the prover, the verifier presents the committed information. Now he takes the other side.

Common Input: The Payley graph (G_2) of order 9 and the bull graph (G_1).

Private Input: An isomorphism ϕ from G_1 to a sub-graph H of G_2 (shown in red).



Prover		Verifier
1. Chooses an isomorphism π: $G_2 \rightarrow G'_2$. 2. Creates an adjacency matrix A for G'_2 3. Sends A to the verifier. $A = \begin{bmatrix} 0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&1&0&1 \\ 1&1&0&1&0&0&1&0&0 \\ 1&0&1&0&1&1&0&0&0 \\ 0&1&0&1&0&1&0&0&1 \\ 0&0&0&1&1&0&1&1&0 \\ 0&1&1&0&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0&1 \\ 1&1&0&0&1&0&0&1&0 \end{bmatrix}$	→	(commitment) $A = \begin{bmatrix} 0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&1&0&1 \\ 1&1&0&1&0&0&1&0&0 \\ 1&0&1&0&1&1&0&0&0 \\ 0&1&0&1&0&1&0&0&1 \\ 0&0&0&1&1&0&1&1&0 \\ 0&1&1&0&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0&1 \\ 1&1&0&0&1&0&0&1&0 \end{bmatrix}$ 4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends π and the decommitment information for A to the verifier.	→ 7. Checks that $\pi(G_2) = G'_2$ and that π is a valid isomorphism.
$c = 1$	6. Sends the decommitment information for the entries of A that correspond to edges of $\pi(\phi(G_1))$ to the verifier. 7. Sends the permutation $\phi\pi$ to the verifier.	→ $A = \begin{bmatrix} 0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&1&0&1 \\ 1&1&0&1&0&0&1&0&0 \\ 1&0&1&0&1&1&0&0&0 \\ 0&1&0&1&0&\mathbf{1}&0&0&1 \\ 0&0&0&1&\mathbf{1}&0&\mathbf{1}&\mathbf{1}&0 \\ 0&1&1&0&0&\mathbf{1}&0&\mathbf{1}&0 \\ 1&0&0&0&0&\mathbf{1}&\mathbf{1}&0&\mathbf{1} \\ 1&1&0&0&1&0&0&\mathbf{1}&0 \end{bmatrix}$ 8. Checks that $\pi(\phi(G_1))$ is isomorphic to G_1, and that $\pi(\phi(G_1))$ is what was revealed.

Figure 4: ZKP2 for the sub-graph isomorphism problem example

Verifier simulates the *verifier*. The verifier guesses randomly and uniformly whether to request the graph or an isomorphic sub-graph. Because the verifier has no way to guess with any advantage whether the committed matrix contains the graph or an isomorphic sub-graph (because the choice is random), there is a 50% chance that he requests an option (graph or sub-graph) that the verifier, in the guise of prover, can supply. If not, the verifier backs up the simulation to the state it was in at the start of this round and restarts the entire round (verifier simulating the *prover*).

End.

In an expected 2 passes through each round, the verifier will obtain the information without the help of the prover. Thus the interaction does not help the verifier do something with the prover in expected polynomial time that he could not as well have done without the prover in expected polynomial time. ■

Consider the zero-knowledge proof system ZKP2 for the sub-graph isomorphism problem. This protocol shows the basic structure of all of the protocols in this section. Figure 5 illustrates the protocol in the case that the prover is attempting to cheat. The prover does not have a valid isomorphism from G_1 to a sub-graph of G_2, and the verifier must catch this.

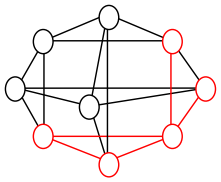
In order for these zero-knowledge proof systems to be of use, we must determine the total number of bits to be transferred. In ZKP2, the graphs that we are considering are simple, undirected graphs. This implies that the adjacency matrices will be symmetric with zeros along the diagonal and with all entries equal to either 0 or 1. Thus in a graph with n vertices, the prover only needs to transmit $\binom{n}{2}$ entries of A to the verifier, where $n = |G|$. Hence step 3 requires the transmission of $\binom{n}{2}$ entries, each of which is one bit. In step 5, the verifier sends one bit. If the verifier chooses $c = 0$, the prover must send the isomorphism π. We can send this in list form, and so we need $n \log_2 n$ bits. If the verifier chooses $c = 1$, the prover must identify the isomorphism $\pi(\phi(G_1))$ and must send the permutation $\phi\pi$ to the verifier. This requires sending decommitment information for the edges corresponding to $\pi(\phi(G_1))$ and also sending a permutation in list form with n entries.

When considering the maximum number of bits that will be necessary in ZKP2 not including what is needed for commitment methods, the number transferred will be:

$$\binom{n}{2} + 1 + n \log_2 n \qquad (6)$$

Common Input: The Payley graph (G_2) of order 9 and the bull graph (G_1).

Private Input: An invalid isomorphism ϕ from G_1 to a sub-graph H of G_2 (shown in red).



Prover		Verifier
1. Chooses an isomorphism $\pi: G_2 \rightarrow G'_2$. 2. Creates an adjacency matrix A for G'_2 3. Sends A to the verifier. $A = \begin{bmatrix} 0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&1&0&1 \\ 1&1&0&1&0&0&1&0&0 \\ 1&0&1&0&1&1&0&0&0 \\ 0&1&0&1&0&1&0&0&1 \\ 0&0&0&1&1&0&1&1&0 \\ 0&1&1&0&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0&1 \\ 1&1&0&0&1&0&0&1&0 \end{bmatrix}$		(commitment) $A = \begin{bmatrix} 0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&1&0&1 \\ 1&1&0&1&0&0&1&0&0 \\ 1&0&1&0&1&1&0&0&0 \\ 0&1&0&1&0&1&0&0&1 \\ 0&0&0&1&1&0&1&1&0 \\ 0&1&1&0&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0&1 \\ 1&1&0&0&1&0&0&1&0 \end{bmatrix}$ 4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends π and the decommitment information for A to the verifier.	7. Checks that $\pi(G_2) = G'_2$ and that π is a valid isomorphism.
$c = 1$	6. Sends the decommitment information for the entries of A that correspond to edges of $\pi(\phi(G_1))$ to the verifier. 7. Sends the permutation $\phi\pi$ to the verifier.	$A = \begin{bmatrix} 0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&1&0&1 \\ 1&1&0&1&0&0&1&0&0 \\ 1&0&1&0&1&\mathbf{1}&0&0&0 \\ 0&1&0&1&0&\mathbf{1}&0&0&1 \\ 0&0&0&\mathbf{1}&\mathbf{1}&0&\mathbf{1}&\mathbf{1}&0 \\ 0&1&1&0&0&\mathbf{1}&0&\mathbf{1}&0 \\ 1&0&0&0&0&\mathbf{1}&\mathbf{1}&0&1 \\ 1&1&0&0&1&0&0&1&0 \end{bmatrix}$ 8. Sees that $\pi(\phi(G_1))$ is not isomorphic to G_1 and so rejects the prover.

Figure 5: ZKP2 for sub-graph isomorphism with cheating prover example

If the maximum amount of information to be transmitted is 10 kilobits, then we must have:

$$\binom{n}{2} + 1 + n \log_2 n \leq 10000 \tag{7}$$

$$n \leq 134 \tag{8}$$

Thus the largest instance that could be considered would have at most 134 vertices in the larger of the two graphs. Note that this maximum occurs given any choice that the verifier makes for c.

3.1.2 Graph Isomorphism Problem

The graph isomorphism problem (GIP) is stated as follows: Given two graphs G_1 and G_2, is there an isomorphism $\phi: G_1 \rightarrow G_2$? The GIP is known to belong to the class NP, but it has not been determined to be NP-complete. It is conjectured that the GIP falls somewhere outside of the classes P and NP-complete (Conte, et al. 2004).

3.1.2.1 Algorithms

The three main algorithms for solving the graph isomorphism problem are Nauty (1981), Ullman's algorithm (1976), and VF2 (2004). All three algorithms are able to solve instances of the problem at remarkable speeds. However, VF2 seems to consistently outperform Ullman's algorithm (Cordella, et al. 2004), so the focus of this section will be on Nauty and VF2.

The Nauty algorithm, created by Brendan McKay (McKay 1981), uses a large amount of group theory to determine a canonical labeling of the graphs (Fortin 1996). The main idea of the algorithm is then centered on the fact that if the labelings of the two graphs are the same, then the graphs must be isomorphic. VF2, on the other hand, relies upon backtracking and pruning the search space according to some specified feasibility rules (Cordella, et al. 2004).

In comparing VF2 and Nauty, neither algorithm clearly outperforms the other. In the results of a set of tests comparing the two algorithms on three different classes of graphs ranging from 20 to 1,000 vertices, Nauty appears to be more effective on random graphs, while VF2 is more effective on 2D-mesh graphs and bounded valence graphs (Cordella, et al. 2004). In further testing, it is shown that on all benchmark classes of graphs that were selected with a maximum of 1100 vertices, at least one of VF2 and Nauty can solve the problem instance in less than one second (Foggia, Sansone and Vento 2001).

While it has not yet been determined which classes of graphs the algorithms Nauty and VF2 struggle with, one idea has appeared in the literature (Fortin 1996), (Hernandez-Goya and Caballero-Gil 2004). It is possible to create hard instances of the GIP by swapping the endpoints of two different edges in a highly symmetric regular graph. It is reported that the resulting graph will be several hundred times harder for Nauty (Fortin 1996).

3.1.2.2 Discussion of Existing Protocols

Given the efficiency of the existing algorithms, the graph isomorphism problem will be difficult to use as a base problem for a secure protocol. However, we will discuss two types of zero-knowledge proof systems for the graph isomorphism problem. The first type of zero-knowledge protocol that exists for the graph isomorphism problem is identical to that of the sub-graph isomorphism problem. The only difference between protocols for the two problems is that the sub-graph is no longer a proper sub-graph, but the entire graph, i.e. $H = G_2$.

The second type of zero-knowledge proof system works only for the graph isomorphism problem. The protocol takes as public input two graphs G_1 and G_2 and as private input an isomorphism ϕ: $G_1 \rightarrow G_2$. First, the prover creates an isomorphic copy G'_2 of G_2 (say $G'_2 = \psi(G_2)$) and sends the copy to the verifier. The verifier chooses a challenge bit and sends that choice to the prover. If the verifier sent a challenge bit equal to zero, then the prover sends ψ to the verifier and the verifier checks that $G'_2 = \psi(G_2)$. If the verifier sent a challenge bit equal to one, then the prover sends $\phi\psi$ to the verifier, who checks that $\psi\big(\phi(G_1)\big) = G'_2$ (Goldreich, Micali and Wigderson 1991), (Hernandez-Goya and Caballero-Gil 2004), (Simari 2002), (Grigoriev and Shpilrain 2008).

As mentioned, the ease with which the current algorithms are able to solve exactly the graph isomorphism problem makes these protocols mostly useless. Unless a class of difficult instances is determined, the protocols are not secure, even though they satisfy the necessary properties for a zero-knowledge proof system.

3.1.3 Graph Clustering Problem

The graph clustering problem (GCP) is a more general case of both the graph isomorphism and the graph non-isomorphism problem (the complement of the graph isomorphism problem). The GCP as defined as follows (Goldreich 1996): Given a sequence of graphs $\{G_1, \dots, G_n\}$, and a sequence of positive integers $\{m_1, \dots, m_k\}$, does there exist a partition $C_1, \dots, C_k$ of $[n]$ such that:

1. $|C_i| = m_i$ for $i = 1, \dots, k$.
2. For all $i \in [k]$ and every $x, y \in C_i$, the graphs G_x and G_y are isomorphic.
3. For all $x \neq y \in [k]$ and all $a \in C_x$ and all $b \in C_y$, the graphs G_a and G_b are not isomorphic.

In other words, we are looking to determine if under the equivalence relation of graph isomorphisms, the sizes of the equivalence classes are represented by the given sequence of positive integers.

3.1.3.1 Existing Zero-Knowledge Proofs

The following noninteractive zero-knowledge protocol for GCP relies upon several foundational theorems. The first theorem states that we can construct a monotone formula that determines the value of $THRESH_{t,n}(x_1, \dots, x_n)$ in polynomial-time, where $THRESH_{t,n}$ is a Boolean function with each x_i being a Boolean variable that returns true if and only if at least t of the n variables x_i are true. The second and third theorems state that there exists a perfect zero-knowledge proof system for all instances of true monotone formulae over statements related to graph (non-)isomorphism (DeSantis, Di Crescenzo and Persiano, et al. 1994).

The zero-knowledge proof system discussed below, published by (DeSantis, Di Crescenzo and Goldreich, et al. 1999), takes as public input a sequence of n graphs and a sequence of k positive integers.

1. The prover P proves that the equivalence relation has at least k equivalence classes.

   Determining that at least $k - 1$ graphs are non-isomorphic to all earlier graphs in the initial sequence proves this statement. To accomplish this, we use the first and third theorems to prove in zero-knowledge that $T_1 = THRESH_{k-1,n-1}(f_2, \dots, f_n)$, where $f_i = (\wedge_{j<i} (G_i \text{ is not isomorphic to } G_j))$.

2. P proves that the equivalence relation has at most k equivalence classes.

   Determining that at most $(n-1) - (k-1)$ graphs are isomorphic to all earlier graphs in the initial sequence proves this statement. To accomplish this, we use the first and second theorems to prove in zero-knowledge that $T_2 = THRESH_{n-k,n-1}(f'_2, \dots, f'_n)$, where $f'_i = (\vee_{j<i}(G_i \text{ is isomorphic to } G_j))$.

3. P proves that at least a certain number of equivalence classes have a given minimum size.

   We first define $s_0 = 0$, $s_1 < \cdots < s_d$ such that $\{s_1, \dots, s_d\} = \{m_1, \dots, m_k\}$, and define $n_i = |\{j : m_j = s_i\}|$ for each $i = 1, \dots, d$. P proves the statements:

   $U_{3,i} =$ 'at least $x = n_d + \cdots + n_{d-i+1}$ classes have size at least $y = s_{d-i+1}$.' for $i = 1, \dots, d$

   This is done by proving in zero-knowledge $T_3 = THRESH_{n-k+x,n}(g_1, \dots, g_n)$ where: $g_i = (\vee_{j<i}(G_i \approx G_j)) \vee (THRESH_{y,n}((G_i \approx G_1), \dots, (G_i \approx G_n)))$.

4. P proves that at least a certain number of equivalence classes have a given maximum size.

   Using the same definitions as in step 3, P proves the statements:

   $U_{4,i} =$ 'at least $x = n_1 + \cdots + n_i$ classes have size at most $y = s_i$.' for $i = 1, \dots, d$

   This is accomplished by proving in zero-knowledge $T_4 = THRESH_{x,n}(g'_1, \dots, g'_n)$ where: $g'_i = (\wedge_{j<i}(G_i \neq G_j)) \vee (THRESH_{n-y,n}((G_i \neq G_1), \dots, (G_i \neq G_n)))$.

To prove that this is in fact a noninteractive zero-knowledge protocol, we note that T_1 and T_2 hold if and only if there are exactly k equivalence classes in the sequence of graphs. Also, through basic algebraic manipulation and induction, it is easily proven that statements T_3 and T_4 hold for every i if and only if the k equivalence classes have the correct sizes as specified by the sequence of positive integers (DeSantis, Di Crescenzo and Goldreich, et al. 1999). Since the protocol is a composition of zero-knowledge protocols based on the $THRESH$ function, the protocol is also zero-knowledge.

3.1.3.2 Discussion of Problem

It does not appear to have been discussed in the literature as to whether GCP is an NP-complete problem or not. It is clear that it lies in the class NP, as given a true instance of the GCP, a witness for the problem is a set defining the partitions together with a set of isomorphisms from each graph to another in the same partition class, and this witness can be easily verified. We do note, however, that when our sequence of positive integers is $\{1, 1\}$ then the problem is an instance of the graph non-isomorphism problem, and when our sequence of positive integers is $\{2\}$ then the problem becomes an instance of the graph isomorphism problem. Thus we can see that GCP is at least as hard as the graph isomorphism and graph non-isomorphism problems, and that determining the complexity of the graph (non-)isomorphism problem will determine the complexity of the graph clustering problem.

While no information has been found yet as far as algorithms for solving the graph clustering problem, it should be noted that the problem can be solved by repeatedly applying any algorithm for solving the graph isomorphism problem. In the worst case, each graph would be in a separate equivalence class. This would then imply that any graph isomorphism algorithm would need to be applied to the instance fewer than $n(n - 1)/2$ times in order to determine the equivalence classes (each graph needs only to be compared to one graph in each equivalence class determined before). Thus in instances where the decision version of the graph isomorphism problems involved can be determined in under t seconds, the entire graph clustering problem instance could be solved in under $n^2 t$ seconds. Due to the significantly increased amount of information needed for the problem, namely the sequence of graphs, the graph clustering problem is most likely not a good candidate for a security protocol.

3.1.4 Independent Set Problem

The independent set problem (ISP) is stated as follows: Given a graph G and an integer k, does G contain an independent set of size k? This question is the decision version of an optimization problem known as the maximum independent set problem. The optimization (maximum) independent set problem is as follows: Given a graph G, what is the size of a maximum independent set in G? The optimization problem is an NP-hard problem and the decision version is a well-known NP-complete problem (Garey and Johnson 1979).

Another NP-complete problem that is equivalent to the ISP is the maximum clique problem (MCP). The MCP is stated as follows: Given a graph $\overline{G}$ and an integer k, does $\overline{G}$ contain a clique of size k? The equivalence of the two problems can be seen clearly when we observe that the complement of an independent set is a clique and vice versa. Thus given any instance of the ISP, we can easily convert it to an instance of the MCP merely by considering the complement of the graph in question.

A problem closely related to the ISP is the k-independent set problem (KIS). The problem is stated as follows: Given a graph G and positive integers j and k, does there exist a k-independent set (a set of vertices such that between any two distinct vertices in the set, the length of the shortest path between them is at least k) of size j? The KIS is an NP-complete problem – a fact that is clearly seen when we observe that the ISP is a subproblem of the KIS (Desmedt and Wang 2003).

3.1.4.1 Algorithms

Several near-optimal algorithms have been proposed to deal with the ISP and the MCP. In a relatively recent publication, it was reported that the most competitive algorithms are DLS (Dynamic Local Search), RLS (Reactive Local Search), and VNS (Variable Neighborhood Search) (Grosso, Locatelli and Pullan 2008). While these algorithms are geared towards the MCP, the equivalence of the MCP and the ISP allows the algorithms to be used easily on either problem.

DLS-MC, a DLS variant, was introduced in 2006 (Pullan and Hoos 2006) and is based on stochastic local search. It assigns penalty values to the vertices in order to help the algorithm avoid cycling around local optima. The creators conclude from their testing that the DLS-MC outperforms several older algorithms and improves upon the previously existing DLS algorithms. The RLS algorithm was improved upon in 2007, and so has been replaced by R-Evo and RLS-Evo. These modified RLS algorithms both begin by obtaining an initial estimate, after which a better solution is searched for. They employ a model-based approach in which the current solution is used to provide information about possible locations of a better solution (Battiti and Brunato 2007). An efficient algorithm that often outperforms RLS is KLS. KLS is based on the technique of variable depth search, a variation of local search, and proceeds by adding and removing vertices from the current clique in order to find a larger one (Katayama, Hamamoto and Narihisa 2005).

The general result of the published material on ISP or MCP algorithms is that there is no "best" algorithm for every instance of the problem. Fortunately, there is a standard set of benchmark graphs that most algorithms are tested and compared on. These benchmarks, known as the DIMACS benchmark instances for the maximum clique problem, originated from The Second DIMACS Implementation Challenge: 1992-1993 (Johnson and Trick 1996). The DIMACS graphs range in size from under 100 vertices to 4,000 vertices, however it is not clear

what determines the difficulty level of the graphs. Reviewing the published results, it appears that almost every DIMACS benchmark graph can be solved for the best known solution in less than 200 seconds.

3.1.4.2 Existing Zero-Knowledge Proofs

The zero-knowledge proof systems for the independent set problem take as public input a graph, G, and a positive integer k, and as private input a set $S \subseteq V(G)$, where S is an independent set. A zero-knowledge proof system (ZKP3) for the ISP is illustrated in Figure 6 (Desmedt and Wang 2003). The prover (P) chooses randomly an isomorphism π to permute the graph G and then sends a commitment to this new graph, $\pi(G)$, to the verifier (V). V then chooses a random bit c, which is sent to P. If $c = 0$, P sends π to V along with the decommitment information for $\pi(G)$, and V checks that $\pi(G)$ was formed correctly from G and π. If $c = 1$, then P sends the decommitment information for $\pi(G[S])$ to V, who checks that $\pi(G[S])$ has all entries equal to zero. This then implies that S is an independent set.

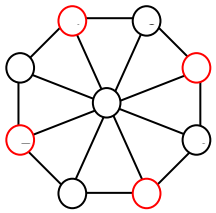
3.1.4.3 Discussion of Existing Protocols

When we consider the soundness property of ZKP3, a cheating prover with a no-instance of the problem will only be caught when the verifier chooses $c = 1$. As the verifier chooses c randomly and uniformly from the set $\{0,1\}$, the probability that a cheating prover will be caught in each round is $1/2$. Thus the probability that a verifier will not reject a cheating prover after k rounds is $\left(\frac{1}{2}\right)^{-k}$, and so the soundness property is satisfied. The protocol ZKP3 also satisfies the completeness and zero-knowledge properties and therefore is a zero-knowledge proof system (Desmedt and Wang 2003).

Because of the equivalence between the independent set problem and the maximum clique problem, ZKP3 can be slightly modified to give a valid zero-knowledge proof system for the MCP. The only change that needs to be made is in the last step that occurs after a verifier chooses $c = 1$. Instead of checking that every transferred matrix entry is zero, the verifier must check that every entry that is not along the diagonal is equal to one. This then demonstrates that the sub-graph revealed is in fact a complete graph, as every pair of distinct vertices has an edge between them.

Common Input: The wheel graph (G) of order 9 (shown) and a positive integer $k = 4$.

Private Input: An independent set of size k (shown in red)



Prover		Verifier
1. Chooses an isomorphism $\pi: G \to G'$. 2. Creates an adjacency matrix A for G' 3. Sends A to the verifier. $A = \begin{bmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&1&1&1&1&1&1&1 \\ 1&1&0&1&0&0&0&0&0 \\ 0&1&1&0&1&0&0&0&0 \\ 0&1&0&1&0&1&0&0&0 \\ 0&1&0&0&1&0&1&0&0 \\ 0&1&0&0&0&1&0&1&0 \\ 0&1&0&0&0&0&1&0&1 \\ 1&1&0&0&0&0&0&1&0 \end{bmatrix}$		(commitment) $A = \begin{bmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&1&1&1&1&1&1&1 \\ 1&1&0&1&0&0&0&0&0 \\ 0&1&1&0&1&0&0&0&0 \\ 0&1&0&1&0&1&0&0&0 \\ 0&1&0&0&1&0&1&0&0 \\ 0&1&0&0&0&1&0&1&0 \\ 0&1&0&0&0&0&1&0&1 \\ 1&1&0&0&0&0&0&1&0 \end{bmatrix}$ 4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends π and the decommitment information for A to the verifier.	7. Checks that $\pi(G) = G'$.
$c = 1$	6. Sends the decommitment information for $\pi(G[S])$ to the verifier.	$A = \begin{bmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&1&1&1&1&1&1&1 \\ 1&1&\mathbf{0}&1&\mathbf{0}&0&\mathbf{0}&0&\mathbf{0} \\ 0&1&1&0&1&0&0&0&0 \\ 0&1&\mathbf{0}&1&\mathbf{0}&1&\mathbf{0}&0&\mathbf{0} \\ 0&1&0&0&1&0&1&0&0 \\ 0&1&\mathbf{0}&0&\mathbf{0}&1&\mathbf{0}&1&\mathbf{0} \\ 0&1&0&0&0&0&1&0&1 \\ 1&1&\mathbf{0}&0&\mathbf{0}&0&\mathbf{0}&1&\mathbf{0} \end{bmatrix}$ 7. Checks that all entries of $\pi(G[S])$ are equal to 0.

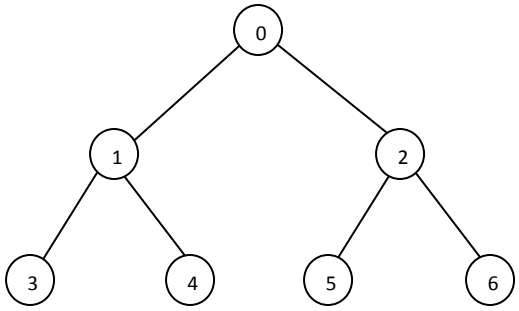
Figure 6: ZKP3 for the independent set problem example

The protocol ZKP3 can also be altered to handle the KIS. First, define the set E_G to be the set of all pairs of vertices (u, v) such that the length of the shortest path between u and v in G is at most $k - 1$, and let $V_G = V(G)$. Then we define $G_k = (V_G, E_G)$. A set $V' \subseteq V(G)$ is a k-independent set in G if and only if V' is an independent set in G_k. By using G_k as the common input to ZKP3, we have a zero-knowledge proof system for the KIS (Desmedt and Wang 2003).

3.1.5 Longest Path Problem

The longest path problem (LPP) is stated as follows: Given a graph G and a positive integer k, does G contain a path of length k? (We are using the assumption that the length of a path is the number of edges of the path.) Figure 7 illustrates an example of the LPP. The more commonly known version of the longest path problem is an optimization problem that asks for a witness for the value $\max\{P \text{ a path}: P \in G\}$. Our phrasing of LPP is merely the decision version that corresponds to the optimization problem. The LPP is an NP-complete problem, and contains the Hamiltonian path problem as a subproblem. However, the LPP is a more difficult problem than the Hamiltonian path problem as the longest path in the graph does not necessarily travel through every vertex. It is an easy reduction from the Hamiltonian path problem to the LPP, and hence the NP-completeness is clear. The optimization version of the problem is NP-hard. There are few graph classes that are known to be easily solvable (in polynomial time). One class of graphs that can be solved quickly is the class of directed acyclic graphs (Garey and Johnson 1979).

Consider the complete binary tree on 7 vertices, as shown.



The maximal paths in the tree are:

1. (3, 1, 4) length = 2
2. (3, 1, 0, 2, 5) **length = 4**
3. (3, 1, 0, 2, 6) **length = 4**
4. (4, 1, 0, 2, 5) **length = 4**
5. (4, 1, 0, 2, 6) **length = 4**
6. (5, 2, 6) length = 2

In this example, the maximum path length is 4. Hence we have:

$$\max_{\substack{P \text{ a path} \\ P \in G}} |P| = 4$$

A witness for this value is the path $P = (3,1,0,2,5)$.

Figure 7: Longest path problem example

3.1.5.1 Algorithms

There are few algorithms that are capable of coping with the LPP. Even approximation algorithms are difficult to come by, as the optimization problem associated with the LPP is thought to lie outside of the class of problems APX - the class of optimization problems for which polynomial-time approximation algorithms with approximation ratios bounded by constants exist (Björklund and Husfeldt 2003). In fact, it has been proven that the longest path problem must lie outside of APX unless P = NP (Karger, Motwani and Ramkumar 1997).

It seems that one of the best performing algorithms currently is a hybrid depth-first-search algorithm that produces either an exact solution to the problem instance in $\tilde{O}(2^{n/\alpha(n)})$ time, where $\tilde{O}\big(g(n)\big) = O(g(n)\log^{k} g(n))$ for some k, or an $O(\alpha(n)\log n)$-approximation, for any α that is an unbounded function (Vassilevska, Williams and Woo 2006). Another possible option is applying a sub-graph isomorphism algorithm to the problem, since the length of the longest path will be available as common knowledge in the zero-knowledge proof system considered for this problem.

3.1.5.2 Establishing a Protocol

Because the LPP is a subproblem of the SGI, we can modify ZKP2 slightly to obtain ZKP4, a zero-knowledge proof system for the longest path problem. The common inputs to the protocol are a graph, G, and a positive integer, k, which represents the length of a longest path in G. The private input is the longest path itself.

Figure 8 illustrates a zero-knowledge proof system, ZKP4, for an instance of the LPP. The prover (P) chooses randomly an isomorphism π to permute the graph G and then sends a commitment to this new graph, $\pi(G)$, to the verifier (V). V then chooses a random bit c, which is sent to P. If $c = 0$, P sends π to V along with the decommitment information for $\pi(G)$, and V checks that $\pi(G)$ was formed correctly from G and π. If $c = 1$, then P sends the decommitment information for $\pi(P)$ to V (where $\pi(P)$ represents the entries corresponding to the edges of the path that is the private input) and V checks that $\pi(P)$ forms a path of the specified length.

Note that in ZKP4 the prover does not need to send any information in addition to the edges of the path to the verifier. The permutations used by the prover are unnecessary information for the verifier, as checking that the edges revealed form a path is a simple task without the knowledge of the permutations. The prover also does not need to identify the vertices that are endpoints on the path, as the verifier can determine these from the revealed entries by examining which rows and columns have one and only one entry equal to 1.

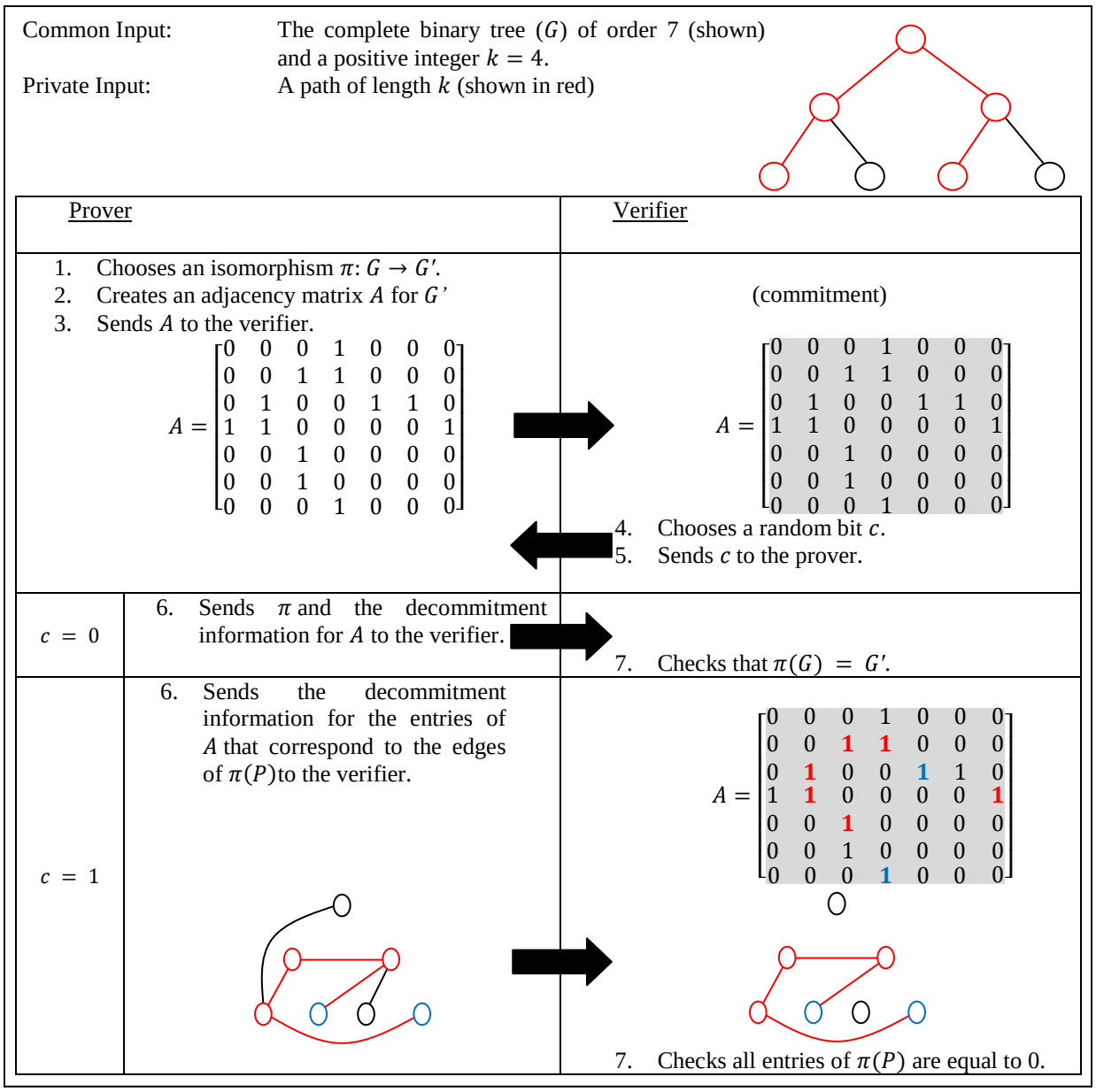


Figure 8: ZKP4 for the longest path problem example

3.1.6 Hamiltonian Cycle Problem

The Hamiltonian cycle problem (HCP) is stated as follows: Given a graph G, does G contain a Hamiltonian cycle (a cycle that passes through every vertex of the graph once and only once)? The HCP is one of the best known NP-complete problems, and is used often in proving other problems to be NP-complete (Garey and Johnson 1979). There are several closely related NP-complete problems, such as the Hamiltonian path problem, the directed Hamiltonian cycle problem, and the Hamiltonian path between two points. Some cases of the HCP are known to be

easy (solvable in polynomial-time), such as if G has no vertex with degree greater than two or if G is a line graph.

3.1.6.1 Traveling Salesman Problem

The HCP has a very well-known subproblem: The Traveling Salesman Problem (TSP). The TSP is stated as follows: Given a graph G' with weighted edges, find a Hamiltonian cycle with the minimum total weight possible. Given an instance of the HCP, it is easy to create an instance of the TSP. Let G be an instance of HCP. Construct G' from G as follows: Let $V(G') = V(G)$ and define the edge set as $E(G') = \{xy: x, y \in V(G')\}$. Assign edge weights as follows: For $e \in E(G')$,

$$w(e) = \begin{cases} 2, & e \in E(G') \backslash E(G) \\ 1, & e \in E(G) \end{cases} \tag{9}$$

If the minimum tour weight of G' is equal to $|V(G)|$, then the graph G has a Hamiltonian cycle. The TSP is an NP-hard optimization problem, and the decision version of the problem (does G' have a tour with total weight less than or equal to some value K) is an NP-complete problem (Garey and Johnson 1979).

3.1.6.2 Algorithms

Since the Hamiltonian cycle problem is a specific case of both the sub-graph isomorphism problem and the traveling salesman problem, any algorithm for solving the SGI or the TSP will also work to solve the Hamiltonian cycle problem. As the TSP is such a well-known and well-researched problem, it is highly likely that the best performing algorithms for the HCP will in fact be TSP algorithms.

A popular TSP algorithm is the Lin-Kernighan (LK) algorithm. The LK algorithm starts with an arbitrary trail that reaches all vertices of the graph (and may include passing through some vertices more than once). It then switches paths on the trail in order to shorten it if possible (Marinakis, Migdalas and Pardalos 2005).

Concorde, an exact algorithm for the TSP, is able to solve optimally 106 out of the 110 instances of the TSP in the TSPLIB[3] (a publicly available library of problem instance for the TSP). Of these instances, the largest involves 15,112 cities (Skiena 2008). It is also reported that for the six instances from TSPLIB with between 1000 and 1200 nodes, an algorithm known as the LKH algorithm (Helsgaun 2000) is able to obtain a solution that is no more than 0.2% from the optimal in less than 20 seconds (Johnson and McGeoch, Experimental Analysis of Herustics for the STSP 2002). Because of this, the instances considered for testing of the algorithms for The Eighth DIMACS Implementation Challenge (2001) did not include any with

[3] Available at http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

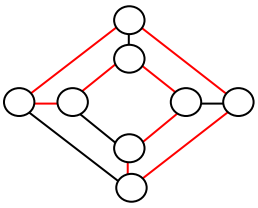
fewer than 1000 nodes (Johnson and McGeoch 2002). These benchmark instances appear to be a mixture of real-world and randomly generated problems.

3.1.6.3 Existing Zero-Knowledge Proofs

The zero-knowledge proof systems for the HCP have as common input a graph, G, that contains a Hamiltonian cycle, and as private input a Hamiltonian cycle, C, in G (note that G may contain more than one Hamiltonian cycle). Figure 9 illustrates a zero-knowledge proof system, ZKP5,

Common Input: The platonic graph of the cube (G) of order 8 (shown)

Private Input: A Hamiltonian cycle (shown in red)



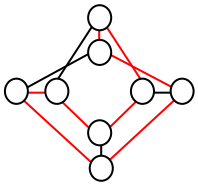
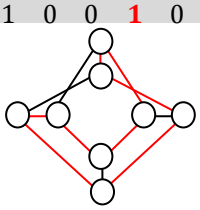
Prover		Verifier
1. Chooses an isomorphism $\pi: G \rightarrow G'$. 2. Creates an adjacency matrix A for G' 3. Sends A to the verifier. $A = \begin{bmatrix} 0&0&0&0&1&1&0&1 \\ 0&0&1&0&1&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&1&0&1&0&0&0&0 \\ 1&1&0&0&0&0&1&0 \\ 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \end{bmatrix}$		(commitment) $A = \begin{bmatrix} 0&0&0&0&1&1&0&1 \\ 0&0&1&0&1&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&1&0&1&0&0&0&0 \\ 1&1&0&0&0&0&1&0 \\ 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \end{bmatrix}$ 4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends π and the decommitment information for A to the verifier.	7. Checks that $\pi(G) = G'$.
$c = 1$	6. Sends the decommitment information for the entries of A that correspond to the edges of $\pi(C)$to the verifier. 	$A = \begin{bmatrix} 0&0&0&0&1&\mathbf{1}&0&1 \\ 0&0&1&0&\mathbf{1}&1&0&0 \\ 0&\mathbf{1}&0&1&0&0&1&0 \\ 0&0&\mathbf{1}&0&1&0&0&1 \\ \mathbf{1}&1&0&1&0&0&0&0 \\ 1&1&0&0&0&0&\mathbf{1}&0 \\ 0&0&1&0&0&1&0&\mathbf{1} \\ 1&0&0&\mathbf{1}&0&0&1&0 \end{bmatrix}$  7. Checks that all entries of $\pi(C)$ are equal to 1 and that $C$is a Hamiltonian cycle.

Figure 9: ZKP5 for the Hamiltonian cycle problem example

for an instance of the HCP (Blum 1986). The prover (P) chooses randomly an isomorphism π to permute the graph G and then sends a commitment to this new graph, $\pi(G)$, to the verifier (V). V then chooses a random bit c, which is sent to P. If $c = 0$, P sends π to V along with the decommitment information for $\pi(G)$, and V checks that $\pi(G)$ was formed correctly from G and π. If $c = 1$, then P sends the decommitment information for $\pi(C)$ to V (where $\pi(C)$ represents the entries corresponding to the edges of the cycle that is the private input) and V checks that $\pi(C)$ forms a Hamiltonian cycle.

The zero-knowledge proof system ZKP5 is just one possible protocol for the HCP. A similar protocol has been published, and the main difference is a reliance on hashing to hide the information that is committed to in ZKP5 (Caballero-Gil and Hernandez-Goya 2006). There is also available a third protocol that assumes that families of collision-free hash functions exist in order to provide a statistical noninteractive zero-knowledge argument with preprocessing (Damgard 1992).

3.1.6.4 Discussion of Existing Protocols

While it may seem tempting to use the same HCP protocol (ZKP5) for the TSP, unfortunately the zero-knowledge property will no longer be satisfied. Because the graph has weighted edges, when the prover reveals the edges of the cycle the verifier will learn information about the edge weights associated with the cycle. For example, if every edge has a different weight in the graph, the verifier will easily be able to identify the TSP tour that is supposed to remain hidden. To create a valid zero-knowledge proof system for the TSP, we must transform the given problem into an instance of the sub-graph isomorphism problem. Given a weighted graph $G = (V, E, W)$ for the TSP (W being the set of edge weights associated with the graph), we construct a new graph G' as follows. For every edge $e \in E(G)$ with $W(e) = k$, we replace e with P_k, a path with k edges (both endpoints of P_k are the endpoints of e). This is illustrated in Figure 10. The problem is now to find a path in the new graph of length equal to the minimum

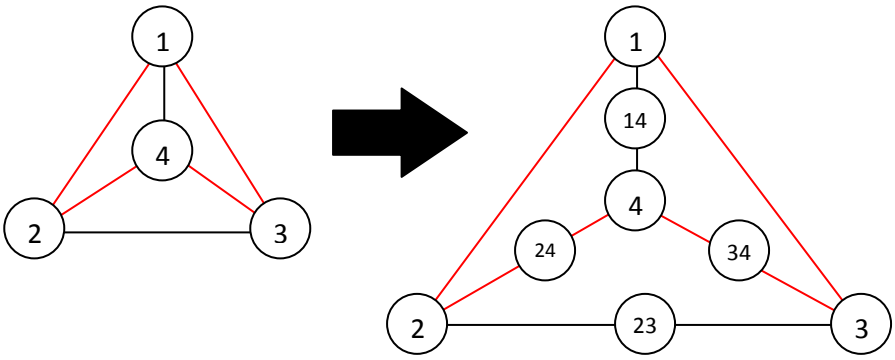


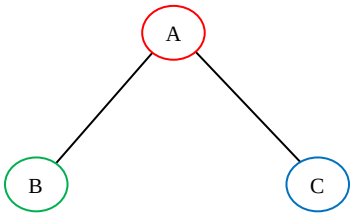
Figure 10: An example of the transformation from the traveling salesman problem to the sub-graph isomorphism problem

TSP weight in the original graph.

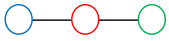
3.1.7 Minimum Bandwidth Problem

The minimum bandwidth problem (MBP) is stated as follows: Given a graph G and a positive integer K, find a linear arrangement of the vertices (a bijective numbering $f: V(G) \to \{1, \ldots, n\}$) such that $\max_{uv \in E(G)} |f(u) - f(v)| \leq K$. This decision problem is an NP-complete problem, and the associated optimization problem (find the minimum value of $\max_{uv \in E(G)} |f(u) - f(v)|$)

Consider the graph shown. We want an ordering of the vertices A, B, and C such that the bandwidth of the ordering is at most 1.



In this example, the possible orderings are:

1. [*A*, *B*, *C*]  Bandwidth = 2
2. [*A*, *C*, *B*]  Bandwidth = 2
3. [*B*, *C*, *A*]  Bandwidth = 2
4. [*B*, *A*, *C*]  **Bandwidth = 1**
5. [*C*, *A*, *B*]  **Bandwidth = 1**
6. [*C*, *B*, *A*]  Bandwidth = 2

So the bandwidth desired is obtained by both ordering 4 and ordering 5.

Figure 11: Minimum bandwidth problem example

is an NP-hard problem (Garey and Johnson 1979). Figure 11 illustrates an example of the MBP.

3.1.7.1 Algorithms

Very few algorithms are able to cope with the MBP efficiently. Both exact and approximate algorithms exist for the problem. As of 2008, the best exact algorithm has time complexity $O(5^n f(|n|))$, where f is a polynomial function (Cygan and Pilipczuk 2008). It seems to be an

open problem as to whether the problem can be solved in $O(2^n f(|n|))$ time, where f is a polynomial function (Woeginger 2003). As for approximation algorithms, as of 2003 the best known approximation algorithm has an $O\left((\log n)^3\sqrt{\log n \log\log n}\right)$ approximation ratio (Woeginger 2003). In 2005, a hybrid algorithm was presented that produces either an ordering that obtains, in $4^{n+o(n)}$ time, the optimal minimum bandwidth or, in polynomial time, an $O((\log^{2.5} n)(\log\log n)(\log^2 \log\log n))$-approximation (Vassilevska, Williams and Woo 2006).

A set of useful benchmark instances are available for the minimum bandwidth problem. The Harwell-Boeing Sparse Matrix Collection (Duff 1992) presents many instances in a range of sizes that originate from real-world applications. While these instances are not generated in any uniform manner, there are several classes that the algorithms all seem to struggle with. For example, the "Cannes" matrices, with instances named can_###, stem from aircraft design. This class of instances appears to be difficult for many algorithms when the order is larger than 200. When the order is greater than 800, as in can_838, most algorithms are unable to solve it exactly (Lim, Rodrigues and Xiao 2006). It would be worth investigating what makes these instances so difficult for the algorithms.

3.1.7.2 Translation to Sub-graph Isomorphism

The minimum bandwidth problem can be viewed as a subproblem of the sub-graph isomorphism problem. Because of this property, any zero-knowledge proof system for the sub-graph isomorphism problem can be applied to the minimum bandwidth problem. Let G be a graph with minimum bandwidth K. Define P_n^K to be the path of length n with additional edges added between every pair of vertices that are at distance at most K apart (on the original path). Then the minimum bandwidth problem can be restated as follows: Given a graph G on n vertices, find an isomorphism $\pi: V(G) \to V(H)$, where $H \subseteq P_{n-1}^K$. The discovered isomorphism from G to a sub-graph of P_{n-1}^K will then give a linear order for $V(G)$ with bandwidth at most K.

For the example illustrated in Figure 3-10, we consider the path P_2^1. In this case, there exists an isomorphism π that maps $V(G)$ to $V(P_2^1)$. One possible mapping is given by:

$$\pi(A) = 2, \pi(B) = 1, \text{ and } \pi(C) = 3 \qquad (10)$$

Thus the isomorphism π gives us a linear ordering identical to ordering number 4 in the example. We may also consider the problem of finding a linear order for $V(G)$ with bandwidth at most 2. In this case, we consider the path P_2^2. It is clear that in this case, any of the orderings illustrated can be mapped isomorphically to a sub-graph of P_2^2.

This process of transforming the MBP to the SGI allows us to employ the same zero-knowledge proof system for the MBP. The common inputs to the protocol are a graph G and the value K of the minimum bandwidth of the graph. The private input is a linear ordering of the vertices that has bandwidth K. The prover permutes the path P_{n-1}^K, and sends a commitment to

this to the verifier. The verifier then chooses randomly whether to check if the isomorphism was constructed correctly or if there is an isomorphic copy of the graph G in P_{n-1}^{K}.

3.1.8 Summary

The sub-graph isomorphism class contains many problems that may be useful as base problems for zero-knowledge proof systems. The minimum bandwidth problem, for example, appears to be a difficult problem with relatively few efficient algorithms to solve it. The same is true of the longest path problem. The Hamiltonian cycle problem or the Hamiltonian path problem may be difficult as well, however the longest path problem intuitively seems harder. The Hamiltonian problems require that all vertices be members of the required cycle or path, whereas in the longest path problem a solver must not only find a path of the required length but must also determine which vertices the path traverses.

One problem that will almost certainly not be useful in creating a secure protocol is the graph isomorphism problem. The current algorithms (Nauty and VF2, for example) are far too efficient at solving large problem instances. In order to create a secure protocol off of the graph isomorphism problem, we would need to use extremely large graphs (over 10,000 nodes), which then dramatically increases the amount of information to be transferred between prover and verifier.

3.2 GRAPH COLORING CLASS

The graph coloring class of problems contains three important problems: graph k-colorability, graph 3-colorability, and equitable 3-colorability. All three problems are NP-complete. Graph 3-colorability is proven NP-complete by a reduction from 3-SAT (a well-known NP-complete subproblem of the satisfiability problem), which then proves the NP-completeness of graph k-colorability. Equitable 3-colorability is proven NP-complete by a reduction from graph 3-colorability easily by adding isolated vertices to a 3-colorable graph to obtain a graph that can be equitably 3-colored. The subproblem structure of the graph coloring class is illustrated in Figure 12.

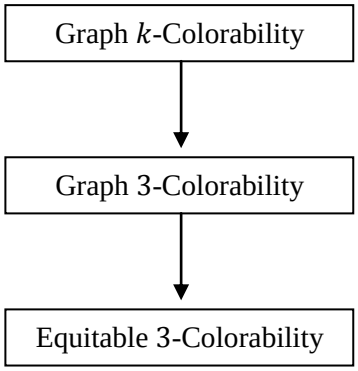


Figure 12: The graph coloring class

3.2.1 Graph Coloring Problem

The graph coloring decision problem is stated as follows: Given a graph G and a positive integer k, is it possible to color the vertices of G with k colors so that every edge has different colored endpoints? More formally, is there a function $f: V(G) \rightarrow [k]$ such that if $xy \in E(G)$, then $f(x) \neq f(y)$? Another alternative is to view the graph k -coloring problem as an optimization-type decision problem. This formulation of the problem is as follows: Given a graph G, partition the vertices into k sets so that m, the number of edges with both endpoints in the same partition class, is minimized. Then G is k-colorable if and only if the minimum value obtained is $m = 0$. The associated chromatic number problem asks for the minimum number of colors needed to color $V(G)$ so that if $xy \in E(G)$ then $f(x) \neq f(y)$. The solution to the problem is the chromatic number of the graph G, and is denoted by $\chi(G)$.

Much work has been done in exploring which classes of graphs have polynomial-time optimal coloring algorithms. For example, the general problem can be solved in polynomial time

for any comparability graph (an undirected graph that is transitively orientable) and any chordal graph (a graph with no induced cycle with length greater than 3) (Golumbic 1980). We also note that when considering a graph with maximum degree at most k, the decision problem becomes trivial due to Brooks' Theorem, which states that $\Delta(G) \geq \chi(G)$ for any graph G that is neither a complete graph nor an odd cycle (Diestel 2006). For the class of random graphs $G(n, p)$, there exist linear-time algorithms for optimal coloring when $p \geq 1.01/n$, where p is the edge probability associated with the random graph (Coja-Oghlan and Taraz 2004).

3.2.1.1 Algorithms

Much work has been done on developing and improving efficient algorithms for the graph coloring problem. While there is an abundance of algorithms focused on achieving near-optimal colorings of a graph, there are very few exact algorithms. However, the near-optimal algorithms can in many cases achieve colorings of a large number of graphs that use the minimum possible number of colors. The algorithms are either based on a local search method, such as tabu search, or on a branch-and-bound-type pruning of the entire search space. Some of the most commonly appearing algorithms in the literature are DSATUR, Tabucol, GH, VNS and Amacol.

Instead of relying on a local search, DSATUR depends on a specific ordering of the vertices. While many improvements have been made to the algorithm, the original method colors the vertices according to the number of colors already present in their neighborhoods. The DSATUR algorithm is continuously being improved upon and is still competitive with the current algorithms (Brélaz 1979). Other similar algorithms based on specific vertex orderings, such as RLF, often appear as a piece of a larger algorithm instead of as a standalone method like DSATUR.

While over 20 years old, Tabucol, a local search algorithm based on tabu search, remains very popular. Tabucol first assigns a random *k*-coloring to the graph, usually with a significant number of conflicting edges (edges with endpoints of the same color). The algorithm then improves this coloring until it has reached the maximum number of iterations allowed (Galinier and Hertz 2006).

The Variable Neighborhood Search algorithm (VNS) is similar to Tabucol, but modifies the searching method. While Tabucol relies on tabu search, VNS uses several neighborhoods in order to avoid getting stuck at local optima (Avanthay, Hertz and Zufferey 2003). Variable Space Search (VSS) is an improvement of VNS. VSS expands upon the idea of considering many neighborhoods to also consider multiple objective functions and search spaces (Hertz, Plumettaz and Zufferey 2008).

A very competitive algorithm is GH, a hybrid evolutionary algorithm, which relies upon a local search method and a crossover function. The crossover function builds a new solution from two previously created partial solutions. GH is quite competitive when it is able to

compute an answer under a given time constraint, however there are many instances where GH does not come up with any solution (Galinier and Hao 1999).

Perhaps the newest algorithm that is worth considering is Amacol. Amacol relies on a central memory solution that contains pieces of solutions and is continuously updated. Using what is currently in the central memory solution, Amacol runs a local search method to improve and create a better solution, and then stores pieces of this new solution (Galinier, Hertz and Zufferey 2008).

While there is no one reference that runs experiments on all four of these algorithms side-by-side, there has been a set of experiments run comparing Tabucol, DSATUR, GH, and Amacol (Galinier, Hertz and Zufferey 2008), another set comparing DSATUR, Tabucol, RLF, and VNS (Galinier and Hertz 2006), and yet another set comparing VSS and Tabucol (Hertz, Plumettaz and Zufferey 2008).

Almost all tests run used a specific benchmark sets of graphs, such as the graphs from The Second DIMACS Implementation Challenge: 1992-1993 (Johnson and Trick, Volume 26: DIMACS Series in Discrete Mathematics and Theoretical Computer Science 1996), that are generally considered to be difficult (meaning that most algorithms struggle to find optimal solutions). In most cases, the DIMACS graphs used contain either 500 or 1000 vertices. Other classes of graphs were used as well, such as the flat graphs (each containing 1000 vertices). VSS was able to obtain an optimal coloring on 16 out of 20 test graphs with a time limit of one hour. In the tests that did not produce optimal colorings, VSS produced colorings using at most five extra colors. Similar results are shown for Tabucol, but with fewer optimal colorings. According to test results from July 2008, GH outperforms all of the other listed algorithms, except for the few cases where it is unable to determine a result under the time constraint (Hertz, Plumettaz and Zufferey 2008).

3.2.1.2 Existing Zero-Knowledge Proofs

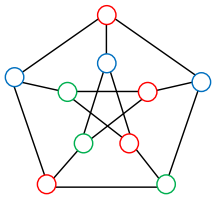
Several zero-knowledge proof systems have been developed for the graph 3-colorng problem (G3C). All proof systems take as public input a graph G and as private input ϕ, a 3-coloring of the vertices of G. In addition to the protocols discussed here, there are a few variations that have been published, such as a protocol that relies upon hiding the coloring through an isomorphic graph (Grigoriev and Shpilrain 2008) and non-interactive zero-knowledge proof systems (Blum, Feldman and Micali 1988), (Kurosawa and Takai 1992).

Figure 13 illustrates ZKP6, a zero-knowledge proof system for G3C (Goldreich, Micali and Wigderson 1991). In the protocol, the prover permutes the coloring of the graph and commits to it before sending it to the verifier. The verifier then selects an edge from the graph randomly and uniformly, sends the edge choice to the prover, and then asks the prover to verify that the edge's endpoints have distinct colors.

A similar zero-knowledge proof system is ZKP7, illustrated in Figure 14. The method of proof is the same, however it is run in parallel instead of sequentially. The prover creates t permutations of the coloring to commit to, while the verifier commits to a set of t edges to challenge the prover with.

Common Input: The Petersen Graph (G) of order 10

Private Input: A 3-coloring ϕ of $V(G)$ (as shown)



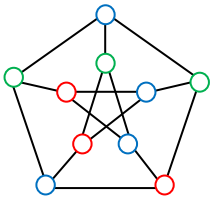
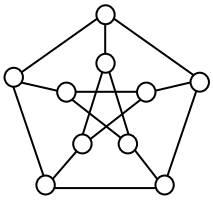
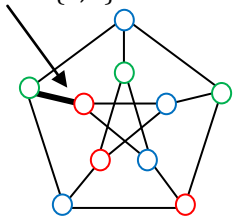
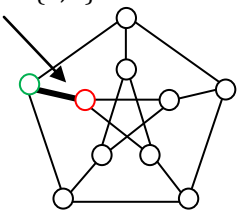
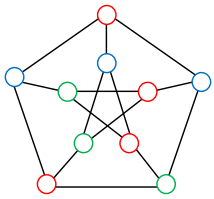
Prover	Verifier
1. Chooses a permutation π of $\{r, b, g\}$. 2. Permutes the coloring of $V(G)$ to obtain a new coloring, $\Psi = \pi(\varphi)$. 3. Sends a commitment to Ψ to the verifier.	
	
	4. Chooses randomly an edge e of the graph. 5. Sends e to the prover.
6. Sends the decommitment information for the endpoints of e to the verifier.	
$e = \{1, 6\}$ 	$e = \{1, 6\}$ 
	7. Checks that the endpoints of e are colored with different colors.

Figure 13: ZKP6 for the graph 3-coloring problem example

Common Input: The Petersen Graph (G) of order 10

Private Input: An 3-coloring ϕ of $V(G)$ (as shown)



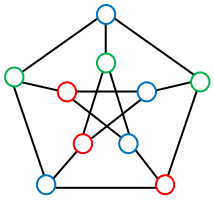
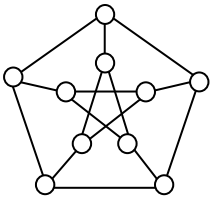
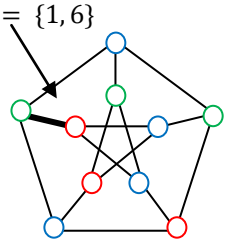
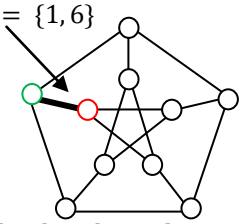
Prover	Verifier
1. Chooses t permutations $\pi_1, \dots, \pi_t$ of $\{r, b, g\}$. 2. Permutes the coloring of $V(G)$ to obtain new colorings, $\Psi_i = \pi_i(\varphi)$ for all $1 \leq i \leq t$. 3. Sends a commitment to each Ψ_i to the verifier.	
	
	6. Chooses randomly a sequence of t edges, $\underline{E} = \{e_i \mid 1 \leq i \leq t\}$, of the graph. 7. Sends a commitment to $\underline{E}$ to the prover.
8. Sends the decommitment information for the endpoints of each e_i corresponding to each Ψ_i to the verifier.	
$e = \{1, 6\}$ 	$e = \{1, 6\}$ 
	9. Checks that the endpoints of each e_i are colored with different colors.

Figure 14: ZKP7 for the graph 3-coloring problem example

3.2.1.3 Discussion of Existing Protocols

If the prover is cheating and does not have a valid 3-coloring of G during ZKP6, then when the prover attempts to 3-color G, the coloring will have at least one edge with both endpoints colored the same. The probability that the verifier will choose an edge that the prover has colored incorrectly can be as low as $1/|E(G)|$. This implies that the probability that the prover will be discovered as a cheater is as low as $1/|E(G)|$, and hence the probability that a cheater will not be discovered in one round could be as high as $1 - 1/|E(G)|$. The number of iterations necessary to achieve a good confidence level in this protocol can therefore be extremely high. To illustrate how bad this probability is, on a graph with only 1,000 edges, we would need to perform 4603 iterations of the protocol in order to achieve a 99% confidence level.

In ZKP7, the probability of catching a cheating prover increases to $(1 - 1/|E(G)|)^t$ in each round. When we take $t = 2n|E(G)|$, this reduces to $(1 - 1/|E(G)|)^t < e^{-n}$. Thus merely by choosing t large enough, it is possible to achieve any desired confidence level in just one round (Goldreich and Kahan 1996).

Now we compute the number of bits to be transferred during the two protocols discussed. In ZKP6, the prover does not need to send an adjacency matrix. Instead, the prover sends an n-element list in which the i^{th} position of the list contains the color of vertex i. Since there are 3 possible colors, each entry requires at most 2 bits to be recognized. The total number of bits needed to transmit the coloring will thus be at most $2n$ (not including commitment). In step 5, the verifier needs to transmit the two vertices that identify each edge selected. If there are n vertices, then to represent a vertex the verifier will need at most $\log_2 n$ bits. Since two vertices must be sent, the verifier will transmit at most $2 \cdot \log_2 n$ bits. Thus not including what is needed for commitment, the total number of bits sent will be:

$$2n + 2 \cdot \log_2 n \tag{12}$$

If the maximum amount of information that can be transmitted is 10 kilobits, then we must have:

$$2n + 2 \cdot \log_2 n \leq 10000 \tag{13}$$

$$n \leq 4987 \tag{14}$$

Hence the largest graph that could be considered would need to have at most 4987 vertices.

It is clear that in ZKP7, the amount of information needed to be transferred increases dramatically from ZKP6. In ZKP7, again the prover will send an n-element list such that the i^{th} position of the list contains the color of vertex i. Since there are 3 possible colors, each entry needs at most 2 bits, and so again (like in ZKP6), the total number of bits to transmit the coloring will be at most $2n$ (not including what is needed for commitments). However in this case there are t different colorings being sent, and so the total number of bits needed is at most $2tn$. In step 5, the prover must transmit the two vertices that identify each edge selected, as before, however the verifier must transmit a list of t edges, requiring that the verifier transmit $2\mathrm{t} \cdot \log_2 n$ bits. Not including what is needed for commitments, the total number of bits sent during ZKP2 will be:

$$2tn + 2\mathrm{t} \cdot \log_2 n \tag{15}$$

If the maximum amount of information to be transmitted is 10 kilobits, then we must have:

$$2tn + 2\mathrm{t} \cdot \log_2 n \leq 10000 \tag{16}$$

In any graph, $|E(G)| \leq \binom{n}{2}$. If we assume, as in the original publication of ZKP2 (Goldreich and Kahan 1996), that $t = 2n|E(G)|$, then we must have that the above inequality simplifies to $n \leq 10$. This tells us that the graphs that we should be considering can have at most 10 vertices, which is not likely to be a very difficult graph coloring instance.

Also, if we desire a 99.99% probability of catching a cheating prover in one round, then we must have (when $t = 2n|E(G)|$):

$$e^{-n} < 0.0001 \quad (17)$$

$$n > 9.21 \quad (18)$$

Thus on problem instances with exactly 10 vertices, we can achieve the desired confidence level in one round while remaining under the upper limit of the number of bits to be transmitted. However, these instances will be solvable quickly and so will not be of use in creating a secure protocol.

3.2.2 Equitable Coloring Problem

The equitable coloring problem (E3C) is formally stated as follows: Given a graph G, color the vertices of G with as few colors as possible such that any two color classes differ in size by at most 1. This problem is NP-complete, and the proof of this is fairly a straightforward reduction from graph coloring.

Note that any equitable coloring algorithm is also a general graph coloring algorithm, and hence the E3C must be at least as hard as the G3C in terms of algorithms finding optimal solutions. Because of this, there are no algorithms to be presented here that were not previously discussed in the general graph coloring section.

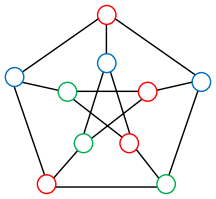
3.2.2.1 Application to Zero-Knowledge Proofs

Note that either of the previously discussed protocols could be applied to the E3C, as the E3C is a subproblem of the G3C. In the zero-knowledge proof systems discussed for the G3C, ZKP6 has a low probability of catching a cheating prover, while ZKP7 has a high amount of information to be transmitted. To address these problems, we turn to the E3C.

A zero-knowledge proof system, ZKP8, is illustrated in Figure 15. The protocol is similar to that of the independent set problem, and takes as public input a graph G and as private input ϕ, an equitable 3-coloring of G. The graph must be committed to as a permuted adjacency matrix to hide the locations of the vertices in each color class, and the coloring must also be committed to (in a list format). The verifier will choose to either check that the permutation was performed correctly or to check that a specified color class induces an independent set in the graph. If the prover has an invalid 3-coloring, then when the verifier requests a color class at least one of the three color classes will not be independent. Thus the probability that the verifier will catch a cheating prover will increase to$1/6$.

Common Input: The Petersen Graph (G) of order 10

Private Input: An equitable 3-coloring ϕ of $V(G)$ (as shown)



Prover		Verifier
1. Chooses a permutation π of $\{r, b, g\}$. Sets $\Psi = \pi\phi$. 2. Chooses a permutation ρ of $V(G)$. 3. Creates an adjacency matrix A for $\rho(G)$. 4. Sends a commitment to A and $\rho\Psi$ to the verifier. $A = \begin{bmatrix} 0&0&0&0&0&1&0&1&1&0 \\ 0&0&1&0&0&1&1&0&0&0 \\ 0&1&0&1&0&0&0&1&0&0 \\ 0&0&1&0&1&0&0&0&1&0 \\ 0&0&0&1&0&1&0&0&0&1 \\ 1&1&0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1&1 \\ 1&0&1&0&0&0&0&0&0&1 \\ 1&0&0&1&0&0&1&0&0&0 \\ 0&0&0&0&1&0&1&1&0&0 \end{bmatrix}$		(commitment) $A = \begin{bmatrix} 0&0&0&0&0&1&0&1&1&0 \\ 0&0&1&0&0&1&1&0&0&0 \\ 0&1&0&1&0&0&0&1&0&0 \\ 0&0&1&0&1&0&0&0&1&0 \\ 0&0&0&1&0&1&0&0&0&1 \\ 1&1&0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1&1 \\ 1&0&1&0&0&0&0&0&0&1 \\ 1&0&0&1&0&0&1&0&0&0 \\ 0&0&0&0&1&0&1&1&0&0 \end{bmatrix}$ 5. Chooses a random bit c. 6. Sends c to the prover.
$c = 0$	7. Sends ρ and A to the verifier.	8. Checks that A was made correctly from ρ.
$c = 1$	9. Sends the decommitment information for entries of A and the entries of $\rho\Psi$ associated with the color class x to the verifier.	7. Chooses $x \in \{r, b, g\}$. 8. Sends x to the prover. $A = \begin{bmatrix} 0&0&0&0&0&1&0&1&1&0 \\ 0&0&1&0&0&1&1&0&0&0 \\ 0&1&\mathbf{0}&1&0&\mathbf{0}&0&1&\mathbf{0}&\mathbf{0} \\ 0&0&1&0&1&0&0&0&1&0 \\ 0&0&0&1&0&1&0&0&0&1 \\ 1&1&\mathbf{0}&0&1&\mathbf{0}&0&0&\mathbf{0}&\mathbf{0} \\ 0&1&0&0&0&0&0&0&1&1 \\ 1&0&1&0&0&0&0&0&0&1 \\ 1&0&\mathbf{0}&1&0&\mathbf{0}&1&0&\mathbf{0}&\mathbf{0} \\ 0&0&\mathbf{0}&0&1&\mathbf{0}&1&1&\mathbf{0}&\mathbf{0} \end{bmatrix}$ 10. Checks that the vertices in the color class x form an independent set.

Figure 15: ZKP8 for the equitable 3-coloring problem example

It is important to note that ZKP8 is only a valid zero-knowledge proof system for the E3C, not for the G3C. If we consider ZKP8 as applied to the G3C, the prover is showing not only that one of the color classes is an independent set, but also the size, k, of the requested color class. The transmission of the size of one color class from the prover to the verifier prevents the protocol from being a zero-knowledge proof system. There is no way that a verifier could have

determined alone that one of the color classes has size k, and so the proof of the zero-knowledge property using simulators would fail.

We must consider now whether ZKP8 is more efficient than either ZKP6 or ZKP7. First, we calculate the total number of bits that must be transferred. Since the graphs we are considering in this example are simple, undirected graphs, the adjacency matrices will be symmetric with zeros along the diagonal and with all entries either 0 or 1. Thus the prover only needs to transmit $\binom{n}{2}$ entries of A to the verifier. The transmission of the coloring needs $2n$ bits. Hence step 4 requires the transmission of $\binom{n}{2} + 2n$ committed entries, each of which is one bit. In step 6, the verifier sends one bit. If $c = 0$, the prover must send the isomorphism π. We can send this in list form, and so we will need $n \log_2 n$ bits. If $c = 1$, the verifier must send an identifier for a color class. Since there are three different color classes, this will require 2 bits. Then the prover must also send the decommitment information for the entries corresponding to edges within the specified color class.

Not including what is needed for commitments, the total number of bits sent will be:

$$\binom{n}{2} + 2n + 1 + n \log_2 n \tag{19}$$

If the maximum amount of information to be transmitted is 10 kilobits, then we must have:

$$\binom{n}{2} + 2n + 1 + n \log_2 n \leq 10000 \tag{20}$$

$$n \leq 133 \tag{21}$$

The largest graph to be considered would need to have at most 133 vertices.

The protocol is more efficient than ZKP6 in terms of the number of rounds necessary to achieve an adequate confidence level. ZKP8 requires 38 iterations to achieve a 99% chance of catching a cheating prover (recall that ZKP6 required 4603 iterations). We also note that if it is possible to use as common input graphs in which it is difficult to produce even two independent color classes, the chance that the verifier can catch a cheating prover increases from $1/6$ to $1/3$.

Thus ZKP8 is a more efficient protocol for E3C than ZKP7, and also gives a better probability of catching a cheating prover than ZKP6 (and ZKP7 depending on what value of t is chosen). All that remains is to prove that ZKP8 is in fact a valid zero-knowledge proof system.

Claim: ZKP8 is a zero-knowledge proof system.

Proof:

Completeness: If the prover has a yes-instance x of E3C, then the verifier will accept x with probability 1.

Soundness: If the prover has a no-instance y of E3C, the prover will be caught only if the verifier chooses $c = 1$, and if the verifier selects a color class that is not independent.

Since c is chosen uniformly and randomly by the verifier, the probability that the verifier will reject y is $1/6$ in each round. This implies that the probability that the verifier does not reject y after c rounds is at most $\left(\frac{5}{6}\right)^c$. When we repeat the protocol for $6k$ rounds, the probability that the verifier does not reject y is $\left(\frac{5}{6}\right)^{6k}$, which is asymptotically close to (and never exceeding) 2^{-k}.

Zero-Knowledge Property: Suppose the verifier is attempting to extract useful information from his conversation with the prover. Then the verifier can, in the same manner, extract the information even without the aid of the prover. In each round he does the following:

Begin.

Verifier simulates the *prover*. The verifier flips a fair coin and, according to the outcome of the coin, commits to either the graph G or an arbitrary 3-partition of n vertices in which each partition class is an independent set. G is committed to in the same way the prover would have done so. The partition is committed to in just the way the prover would have committed to such a partition in G. Then, acting as prover, he presents the committed information to the verifier. Now he takes the other side.

Verifier simulates the *verifier*. The verifier guesses randomly and uniformly whether to request a graph or a partition. Because the verifier has no way to guess with any advantage whether the committed matrix contains a graph or a partition (because the choice is random), there is a 50% chance that he requests an option (graph or partition) that the verifier, in the guise of prover, can supply (in all cases). If a partition was requested but a graph had been committed to, then the verifier guess randomly and uniformly which color class to request. Then there is a 67% chance that the verifier, in the guise of prover, can supply what was requested correctly. This gives a total chance of 83% that the verifier, in the guise of the prover, can supply what is requested. If what is requested cannot be supplied, the verifier backs up the simulation to the state it was in at the start of this round and restarts the entire round (verifier simulating the *prover*).

End.

In an expected 6 passes through each round, the verifier will obtain the information without the help of the prover. Thus the interaction does not help the verifier do

something with the prover in expected polynomial time that he could not as well have done without the prover in expected polynomial time. ■

3.2.3 Summary

The graph coloring problem and equitable coloring problem have positive and negative attributes in terms of zero-knowledge proof systems. A positive feature of these problems is the difficulty level. There exist difficult instances of the problems, and methods have been published on how to create difficult instances. This would provide a strong foundation for a zero-knowledge proof system. The negative aspects of the coloring problems are the soundness probabilities of the proof systems. Compared to the soundness probability of $\frac{1}{2}$ that we see in the sub-graph isomorphism problem and sub-problems, equitable coloring is able to achieve only a soundness probability of $\frac{1}{6}$. This means increasing the number of rounds from 7 to 38 in order to achieve a 99% probability of catching a cheating prover. Given the scenarios in which we are looking to employ zero-knowledge proof systems, it is not realistic to expect that 38 rounds of one protocol will be possible. In order to utilize graph coloring or equitable coloring, we first need to develop a better zero-knowledge proof system.

3.3 OTHER NP-COMPLETE PROBLEMS

3.3.1 Satisfiability

The satisfiability problem (SAT) was the first problem to be proven NP-complete (Garey and Johnson 1979). The problem falls under the category of propositional logic, and is stated as follows: Given a set of Boolean variables and a collection of clauses over the set of variables, is there a truth assignment for the variables such that every clause in the collection is satisfied?

3.3.1.1 Algorithms

Much work has been done on developing algorithms to quickly and efficiently solve instances of the SAT problem. The algorithms fall into two distinct categories: complete algorithms and incomplete algorithms. Incomplete algorithms are stochastic local search based, and are often faster, however fail to prove when an instance of SAT is unsatisfiable. Some well-known incomplete algorithms are WalkSAT and GSAT. Complete algorithms are systematic search algorithms and usually run slower, but are able to determine when an instance of SAT is unsatisfiable. Some complete algorithms that are used often are DPLL, SATO, and GRASP.

One of the first algorithms published was the Davis-Putnam-Logemann-Loveland (DPLL) algorithm. The algorithm is still favored today, and many newer algorithms such as GRASP (Marques-Silva and Sakallah 1999), SATO (Zhang 1997), and BerkMin (Goldberg and Novikov 2002) were created with the same basic idea but with some modifications and

improvements. The main ideas of the systematic search algorithms are backtracking and pruning the search space.

The incomplete algorithms available are also quite efficient in satisfiable instances. When comparing the popular algorithms WalkSAT and GSAT, it appears that neither algorithm is able to outperform the other consistently (Hoos and Stutzle 2000). A new variant of WalkSAT, gNovelty$^+$, has been developed recently for the annual SAT competition, and appears to perform well in the random SAT area (Jia 2007).

A valuable resource for determining runtime performance of the most up-to-date SAT solvers is the annual International SAT Competition[4]. Considering the results of the 2008 SAT Competition, the solvers were given 100 instances of SAT, some of which were unsatisfiable (so only complete algorithms competed). The solvers were allowed 900 seconds (15 minutes) to solve each instance (or determine the instance unsatisfiable) before timing out. The instances consisted of anywhere from 286 to 11,483,525 variables and from 1742 to 32,697,150 clauses. The instances were taken from several benchmark suites, as well as instances from past SAT Competitions (which includes random instances). The first place winner, MiniSat 2.1, was able to solve 81 out of the 100 instances correctly, and the top four winners all solved more than 75 out of the 100 instances correctly.

3.3.1.2 Existing Zero-Knowledge Proofs

There seems to be less work done on the SAT problem with regard to zero-knowledge proof systems than for some other NP-complete problems like graph 3-colorability or sub-graph isomorphism. A zero-knowledge proof system for the SAT problem, ZKP9, which is illustrated in Figure 16, takes as common input a set U of Boolean variables and a collection C of clauses and as private input a set of true/false assignments for the variables in U.

The prover (P) constructs the circuit of truth tables that corresponds to the instance of the problem. P then randomly permutes the rows of each truth table, and randomly complements the columns of the tables, except for the last column of the last table. P then sends a commitment to the permuted and complemented set of tables to the verifier (V). V chooses a random bit c and sends c to P. If V sends $c = 0$ to P, then P sends the decommitment information for all of the truth tables to V. P also notifies V as to which columns were complemented. V then checks that the truth tables were constructed correctly. If V sends $c = 1$ to P, then P sends to V the decommitment information for only the rows that correspond to a satisfying truth assignment before complementing took place. V then checks that these rows lead to a final output of true (and hence explains why the last column of the final truth table cannot be complemented) (Brassard, Chaum and Crepeau 1988).

[4] Available at: http://www.satcompetition.org/

Few other zero-knowledge proof systems have been published for the SAT problem. Papers have been published on non-interactive zero-knowledge proof systems (Damgard 1992), zero-knowledge proof systems with two provers (Dwork, et al. 1992), and an interactive zero-knowledge proof system that focuses on a more secure commitment method than in the protocol presented here (Brassard and Crepeau 1986). While these other protocols may vary slightly from the one illustrated in this report, the extent to which the SAT problem has been studied prevents the problem from being a secure base problem in a protocol.

Common Input: A set U of Boolean variables and a collection C of clauses.

Private Input: A set of assignments for U such that every clause in C is satisfied.

Example Instance:

U = { a, b, c }

C = { (a ∨ ¬b) ∧ (¬a ∨ c) }

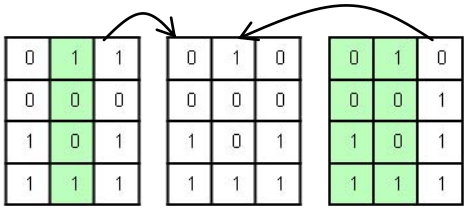
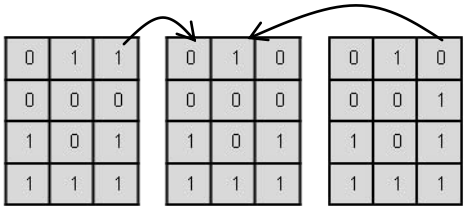
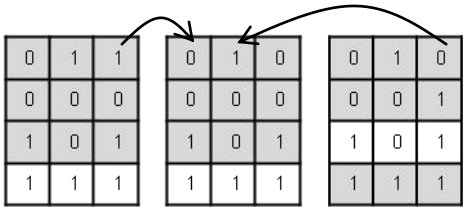
Prover		Verifier
1. Randomly permutes the rows of the truth tables. 2. Randomly complements the columns of the truth tables except for the last column in the final table. (Shown in green) 3. Sends a commitment to the scrambled tables to the verifier.		 4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends the decommitment information for all truth table entries to the verifier and identifies which columns were complemented.	7. Checks that the truth tables were formed correctly.
$c = 1$	6. Sends the decommitment information for the row in each truth table that corresponds to the satisfying truth assignment.	 7. Checks that the sequence of rows revealed outputs true.

Figure 16: ZKP9 for the satisfiability problem example

3.3.1.3 Discussion of Existing Protocols

In discussing the amount of information transferred in the protocol, we will consider an instance of 3-SAT, as any SAT instance can be transformed to a 3-SAT instance. In an instance of 3-SAT, each clause has three variables. The truth table for each clause will have $2^3 = 8$ rows and 4 columns, giving a total of 32 entries. Since each entry is either true or false, we need only 32

bits to send the entries for each clause. If there are $|C|$ clauses total, then in step 3 the prover needs $32|C|$ bits total to send the truth tables (not including the bits needed for the commitment process). In step 5, the verifier sends 1 bit. If $c = 0$, then the prover must send the identifiers for each column in each truth table that is complemented. Since each truth table has 4 columns, at most $4|C| - 1$ columns can be complemented. To send a list of numbers representing the columns that are complemented, the prover must transfer $(4|C| - 1)\log_2(4|C| - 1)$ bits. If $c = 1$, then the prover must reveal one row from each truth table by sending the appropriate decommitment information.

When considering the maximum number of bits that will be necessary in the zero-knowledge proof system illustrated (not including what is needed for commitment), the number transferred will be:

$$32|C| + 1 + (4|C| - 1)\log_2(4|C| - 1) \tag{22}$$

If the maximum amount of information to be transmitted is 10 kilobits, then we must have:

$$32|C| + 1 + (4|C| - 1)\log_2(4|C| - 1) \leq 10000 \tag{23}$$

$$|C| \leq 145 \tag{24}$$

Thus the largest instance of 3SAT that could be considered would have at most 145 clauses. Considering the efficiency of the SAT competition solvers, the instances that would be allowed under this information restriction would not create secure protocols.

3.3.2 Graph Partitioning Problem

The graph partitioning problem (GPP) is stated as follows: Given a graph G and positive integers k and M, is there a partition of the vertices into k equal-sized classes so that there are at most M edges with endpoints in different partition classes? In general, the GPP can consider both weighted and unweighted graphs. The GPP is an NP-complete problem in both the general case (allowing weighted vertices and edges) and in the case restricted to unweighted graphs (Garey and Johnson 1979).

3.3.2.1 Algorithms

There are several algorithms created to solve the GPP. One of the most well-known algorithms was developed in the 1970's and is the Kernighan-Lin (KL) algorithm (Kernighan and Lin 1970). The KL algorithm begins with an initial partition, and then improves it by swapping vertices between the partition classes. This will method will clearly find terminate with a local minimum. However, by allowing swaps of multiple vertices at a time, the algorithm is able to avoid getting trapped at a local minimum, and so it is able to get closer to obtaining a partition that will achieve global minimum.

Another useful algorithm is JOSTLE, a multilevel paradigm algorithm. JOSTLE and other multilevel paradigm algorithms group the graph's vertices to make clusters that then become vertices in a new graph. This can be done by contracting edges. The process is repeated until a smaller graph is obtained, and then existing exact GPP algorithms are applied to the new graph. By expanding and refining the partition of the smaller graph, this algorithm works backwards to create a partition for the original graph (Banos, et al. 2003).

There are many other algorithms for solving the GPP, included an isoperimetric algorithm (Grady and Schwartz 2006), a lock-gain based algorithm (Kim and Moon 2004), greedy algorithms, evolutionary search methods, genetic algorithms (Bui and Moon 1996), simulated annealing algorithms (Johnson, Aragon, et al. 1989), and tabu search methods. As of 2007, JOSTLE appears to be the best performing algorithm for the GPP (Loureiro and Amaral 2007).

Chris Walshaw, of the University of Greenwich, maintains "The Graph Partitioning Archive[5]". The archive consists of a set of benchmark problems, most of which are obtained from real-world applications. Most of the recent publications compare algorithms based on the instances provided there. Considering these instances and more from other sources, it appears that there are some difficult cases of the GPP. Some of these instances as of 2005 were taking over 4 hours to compute (Felner 2005).

3.3.2.2 Establishing a Protocol

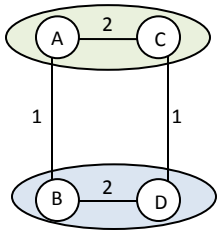
At first glance, it appears to be a simple matter to create a zero-knowledge proof system for the general version of the GPP (on weighted graphs). There are three things that must be proven to the verifier: (1) the partition is valid (every vertex is in one and only one partition class), (2) every partition class contains exactly k vertices, and (3) there are M edges between the partition classes.

Considering these requirements, we arrive at Protocol A, illustrated in Figure 17. Protocol A requires the prover to send a commitment to a permuted adjacency matrix for the graph, and then to prove, at the request of the verifier, either that the permutation was performed correctly or that there exists a partition of the vertices that obtains the required cut cost.

[5] Available at: http://staffweb.cms.gre.ac.uk/~wc06/partition/

Common Input: A graph G (shown), the number of desired partitions ($k = 2$), and the cost of the desired cut ($c = 2$).

Private Input: The partition $(\{A, C\}, \{B, D\})$ that achieves the desired cut cost (as shown).



Prover		Verifier
1. Creates a permutation π of $V(G)$. 2. Creates an adjacency matrix A for $\pi(G)$. 3. Sends a commitment to A to the verifier. $A = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \\ 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix}$	→ (commitment)	$A = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \\ 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix}$
	←	4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends π and the decommitment information for A to the verifier. →	7. Checks that A was formed correctly.
$c = 1$	6. Sends the decommitment information for the cut edges to the verifier. 7. Sends the decommitment information for the non-edges between partition classes to the verifier. →	$A = \begin{bmatrix} 0 & \mathbf{0} & 2 & \mathbf{1} \\ \mathbf{0} & 0 & \mathbf{1} & 2 \\ 2 & \mathbf{1} & 0 & \mathbf{0} \\ \mathbf{1} & 2 & \mathbf{0} & 0 \end{bmatrix}$ 8. Checks that the total sum of edges shown is equal to twice the desired cut cost. 9. Checks that the partition classes have equal size and that there is the correct number of classes. $A = \begin{bmatrix} 0 & \mathbf{0} & 2 & \mathbf{1} \\ \mathbf{0} & 0 & \mathbf{1} & 2 \\ 2 & \mathbf{1} & 0 & \mathbf{0} \\ \mathbf{1} & 2 & \mathbf{0} & 0 \end{bmatrix}$ ← There are 2 nodes in one class ← There are 2 nodes in the other class ← The two classes are distinct and disjoint

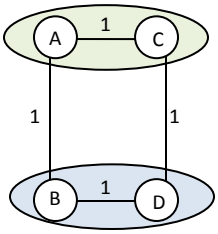
Figure 17: Protocol A for the graph partitioning problem example

While Protocol A satisfies the completeness and the soundness properties that are required for any interactive proof system, it does not satisfy the zero-knowledge property and hence is not able to qualify as a zero-knowledge proof system. In steps 6 and 7, the prover opens all entries that correspond to edges between partition classes. This then tells the verifier how many edges there are between the partition classes, and what the different weights are (but not which vertices the edges are between). This is information that the verifier could not possibly have determine alone without the help of the prover.

A modification to Protocol A is shown in Protocol B. Using the same protocol but requiring that the common input is a graph with all edge weights equal to 1, the prover is not revealing any new information. Because the cost of the cut is common knowledge, the verifier already knows how many edges the graph has between partition classes (provided that all edge-weights are equal to 1). Protocol B is illustrated in Figure 18. The modification from weighted to unweighted graph now allows the protocol to be a zero-knowledge proof system.

Common Input: An unweighted graph G (shown), the number of desired partitions ($k = 2$), and the cost of the desired cut ($c = 2$).

Private Input: The partition $(\{A, C\}, \{B, D\})$ that achieves the desired cut cost (as shown).



Prover		Verifier
1. Creates a permutation π of $V(G)$. 2. Creates an adjacency matrix A for $\pi(G)$. 3. Sends a commitment to A to the verifier. $A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$	→	(commitment) $A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$
	←	4. Chooses a random bit c. 5. Sends c to the prover.
$c = 0$	6. Sends π and the decommitment information for A to the verifier. →	7. Checks that A was formed correctly.
$c = 1$	6. Sends the decommitment information for the cut edges to the verifier. 7. Sends the decommitment information for the non-edges between partition classes to the verifier. →	$A = \begin{bmatrix} 0 & \mathbf{0} & 1 & \mathbf{1} \\ \mathbf{0} & 0 & \mathbf{1} & 1 \\ 1 & \mathbf{1} & 0 & \mathbf{0} \\ \mathbf{1} & 1 & \mathbf{0} & 0 \end{bmatrix}$ 8. Checks that the total sum of edges shown is equal to twice the desired cut cost. 9. Checks that the partition classes have equal size and that there is the correct number of classes. $A = \begin{bmatrix} 0 & \mathbf{0} & 1 & \mathbf{1} \\ \mathbf{0} & 0 & \mathbf{1} & 1 \\ 1 & \mathbf{1} & 0 & \mathbf{0} \\ \mathbf{1} & 1 & \mathbf{0} & 0 \end{bmatrix}$ ← There are 2 nodes in one class ← There are 2 nodes in the other class The two classes are distinct and disjoint

Figure 18: Protocol B for the graph partitioning problem example

Claim: Protocol B is a zero-knowledge proof system for the GPP.

Proof:

Completeness: If the prover has a yes-instance x of GPP, then the verifier will accept x with probability 1.

Soundness: If the prover has a no-instance y of GPP, the prover will be caught only when the verifier chooses $c = 1$. Since c is chosen uniformly and randomly by the verifier, the probability that the verifier will reject y is $1/2$ in each round. This implies that the probability that the verifier does not reject y after c rounds is at most $\left(\frac{1}{2}\right)^c = 2^{-c}$.

Zero-Knowledge Property: Suppose the verifier is attempting to extract useful information from his conversation with the prover. Then the verifier can, in the same manner, extract the information even without the aid of the prover. In each round he does the following:

Begin.

Verifier simulates the *prover*. The verifier flips a fair coin and, according to the outcome of the coin, commits to either the graph G or an arbitrary k-partition of n vertices with the correct total cut cost. G is committed to in the same way the prover would have done so. The partition is committed to in just the way the prover would have committed to such a partition in G. Then, acting as prover, he presents the commitment information to the verifier. Now he takes the other side.

Verifier simulates the *verifier*. The verifier guesses randomly and uniformly whether to request a graph or a partition. Because the verifier has no way to guess with any advantage whether the committed matrix contains a graph or a partition (because the choice is random), there is a 50% chance that he requests an option (graph or partition) that the verifier, in the guise of prover, can supply. If not, the verifier backs up the simulation to the state it was in at the start of this round and restarts the entire round (verifier simulating the *prover*).

End.

In an expected 2 passes through each round, the verifier will obtain the information without the help of the prover. Thus the interaction does not help the verifier do something with the prover in expected polynomial time that he could not as well have done without the prover in expected polynomial time. ■

While we have now proven that Protocol B is a zero-knowledge proof system, we must also determine the level of difficulty of the GPP given the amount of information that is revealed in the problem. The modification to the protocol from weighted to unweighted graphs does not

affect the NP-completeness of the problem, as discussed previously. However, many entries of the adjacency matrix have been revealed and could possibly make it easy for an eavesdropper to solve the problem instance.

Consider Protocol B, as illustrated in Figure 5-3. If too many entries must be revealed by the prover, then the isomorphism may be discovered easily by the verifier using an effective graph isomorphism algorithm. Let G be a graph with$|G| = n$, $\|G\| = m$, $|\wp| = k$ and $|P| = \frac{n}{k}$ for $P \in \wp$, where $\wp$ is the partition (as in the set of partition classes). For the verifier to check that the prover's answer is valid, the verifier must see the adjacency matrix entries for all edges and nonedges between partition classes. Thus the number of entries that will be revealed to the verifier is:

$$k \cdot \frac{n}{k} \cdot \left((k-1)\frac{n}{k}\right) = n^2 \cdot \frac{(k-1)}{k} \tag{25}$$

From this, we can see that the minimum possible number of entries that need to be revealed is $n^2/2$, as is the case in the example above when $k = 2$, but that the maximum possible number of entries can be as high as $n(n-1) = n^2 - n$, in the case where $k = n$. It is important to note that the number of entries revealed is entirely dependent on k when the graph G is fixed, and also that the problem does not appear to increase in difficulty when k is increased.

In determining how useful Protocol B is, we must consider the number of bits to be transferred in each round. Since the graphs we are considering in this example are simple, undirected graphs, the adjacency matrices will be symmetric with zeros along the diagonal and with all entries either 0 or 1. Thus the prover only needs to transmit $\binom{n}{2}$ entries of A to the verifier. Hence step 3 requires the transmission of $\binom{n}{2}$ committed entries, each of which is one bit. In step 5, the verifier sends one bit. If $c = 0$, the prover must send the isomorphism π. We can send this in list form, and so we will need $n \log_2 n$ bits. If $c = 1$, the prover must send the decommitment information as specified in the protocol.

Adding everything up and not including what is needed for commitment, the total number of bits sent will be:

$$\binom{n}{2} + 1 + n \log_2 n \tag{26}$$

If the maximum amount of information to be transmitted is 10 kilobits, then we must have:

$$\binom{n}{2} + 1 + n \log_2 n \leq 10000 \tag{27}$$

$$n \leq 134 \tag{28}$$

The largest graph to be considered could have at most 134 vertices under the given restriction.

3.3.3 Minimum Label Spanning Tree

The minimum label spanning tree (MLSTP) is stated as follows: Given a graph G with labeled edges, find a minimum spanning tree that uses the fewest number of labels possible. In other words, given a graph $G = (V, E, l)$, with vertex set V, edge set E, and edge label set l, find an acyclic connected sub-graph $T \subseteq G$ such that $|L_T|$ is minimized, where $|L_T| = \{c \in l: \exists e \in E(T) \text{ with } l(e) = c\}$. In the example graph in Figure 19, $V = \{a, b, c, d\}$, $E = \{uv: u, v \in V\}$, and $l = \{1,2\}$. There are many spanning trees to consider in the graph shown. It is clear that to include vertex d in the spanning tree, at least one edge with label 2 must be included. Thus a minimum labeling spanning tree is $T = (V_T, E_T, l_T)$, where $V_T = \{a, b, c, d\}$, $E_T = \{da, db, dc\}$, and $l_T = \{2\}$. This tree is shown in red in the figure. We should note that in the example illustrated it is clear where the spanning tree lies in the original graph, as only one vertex has three incident edges with the same label. In order to make the private input as safe as possible, it is important to distribute the edge labels as consistently as possible.

The minimum label spanning tree problem is an NP-complete problem when we rephrase it as a decision problem. In fact, it has been proven that no polynomial-time approximation algorithm with a constant approximation ratio can exist unless P = NP. The MLSTP has many real-world applications, such as communications networks. These kinds of networks can use several different types of communications mediums, such as cable, telephone lines, etc. Solving the MLSTP can give a spanning network using as few different mediums as possible (Chang and Leu 1997).

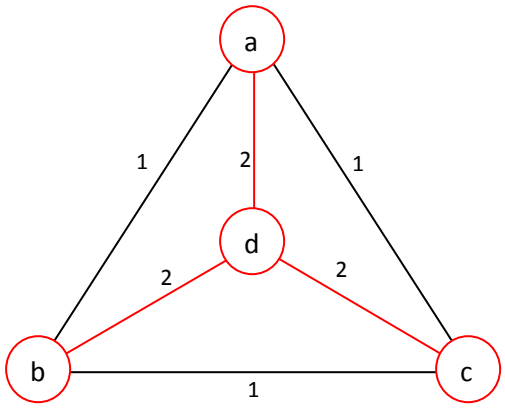


Figure 19: An example of the minimum label spanning tree problem

3.3.3.1 Algorithms

There are several popular algorithms for solving the MLSTP. The most popular algorithm until 2005 was MVCA, the maximum vertex covering algorithm (Consoli, The Development and Application of Metaheuristics for Problems in Graph Theory: A Computational Study 2008). The MVCA was introduced in the paper that first described the MLSTP (Chang and Leu 1997). This approximation algorithm produces a solution that is no greater than $(1 + 2\log n)$ times the optimal. It has also been proven that for any graph with label frequency bounded by some value b, the worst-case bound of MVCA is $H_b = \sum_{i=1}^{b} \frac{1}{i}$, the b^{th} harmonic number (Xiong, Golden and Wasil, Worst-Case Behavior of the MVCA Heuristic for the Minimum Labeling Spanning Tree Problem 2005).

Another algorithm that appears frequently in the literature is a metaheuristic algorithm called the Pilot Method. The Pilot Method improves upon another heuristic algorithm (such as MVCA) using repetition and a look-ahead strategy (Voß and Duin 2003). While the Pilot Method will perform at least as well as the heuristic algorithm that it implements (if not better), it is often quite time consuming because of its repetitive nature.

Other algorithms for the MLSTP include genetic algorithms (Xiong, Golden and Wasil, A One-Parameter Genetic Algorithm for the Minimum Labeling Spanning Tree Problem 2005), tabu search algorithms, and a more recent hybrid algorithm. It appears that the best performing algorithms are VNS (Variable Neighborhood Search) and GRASP (Greedy Randomized Adaptive Search Procedure), which were introduced in 2009 (Consoli, Draby-Downman, et al. 2009). There is also a set of benchmark instances that are maintained by Sergio Consoli[6].

3.3.3.2 Creating a Zero-Knowledge Proof System

Consider the interactive proof system that is illustrated in Figure 20. While the protocol satisfies the completeness and soundness properties of an interactive proof system, it does not satisfy the zero-knowledge property. In the prover's final step, the edges corresponding to the spanning tree are revealed. If $|L_T| > 1$, then the verifier learns how many edges have the same labels. While the verifier does not know which group of edges corresponds to which label, the prover is still transmitting information that the verifier could not have discovered using a simulator.

The next logical question to consider is whether we can restrict the spanning tree so that $|L_T| = 1$ in order to satisfy the zero-knowledge property. Since all trees have $n - 1$ edges, the verifier would then already know how many edges have the same label. However, the problem then becomes too easy to base a secure protocol on. For example, consider the basic algorithm illustrated in Figure 21. If we use any efficient algorithm for finding a spanning tree (MinSpanTree), most of which run in polynomial time, then the problem is easily solvable in an

[6] Available at: http://www.sergioconsoli.com/MLSTP.htm

efficient manner. Thus in order to use the MLSTP, we must first develop a better zero-knowledge proof system.

Common Input: A labeled graph G (shown in Figure 6-4), and the number of distinct labels in a min. label spanning tree ($|L_T| = 1$ in the example).

Private Input: The min. label spanning tree, T (shown in Figure 4 in red).

Prover		Verifier
1. Creates a permutation π of $V(G)$. 2. Chooses a permutation ϕ of the set of labels. 3. Creates an adjacency matrix A for $\pi(\phi(G))$. 4. Sends a commitment to A to the verifier. $A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 2 \\ 1 & 2 & 2 & 0 \end{bmatrix}$	→	(commitment) $A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 2 \\ 1 & 2 & 2 & 0 \end{bmatrix}$
	←	5. Chooses a random bit c. 6. Sends c to the prover.
$c = 0$	7. Sends π and the decommitment information for A to the verifier. →	8. Checks that A was formed correctly.
$c = 1$	7. Sends the decommitment information for the entries of A corresponding to edges in T. →	$A = \begin{bmatrix} 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & 0 & 2 & 2 \\ \mathbf{1} & 2 & 0 & 2 \\ \mathbf{1} & 2 & 2 & 0 \end{bmatrix}$ 8. Checks that the entries correspond to a spanning tree using $\|L_T\|$ labels.

Figure 20: An interactive proof system for the minimum label spanning tree problem example

```
for i ∈ l(G)
    E_i = {e ∈ E(G): l(e) = i}
    G_i = (V(G), E_i)
    if G_i is connected
         return MinSpanTree(G_i)
         exit                  // Exit both loops
    end if
end for
```

Figure 21: An algorithm for the minimum label spanning tree problem with one label

Any zero-knowledge proof system for the MLSTP needs to check the following facts:

(1) $|L_T|$ labels are used on T

(2) T is acyclic
(3) T is connected
(4) T is spanning

It is possible to check (3) and (4) simultaneously by having the verifier request two vertices and requiring the prover to show a path in T between those two vertices. However this will give the verifier information on the spanning tree that could not have been obtained without the help of the prover. The verifier can check (2) by requesting that the prover show that $\|T\| = n - 1$, where n is the number of vertices in G. Again, we arise at the problem of determining how the prover can reveal the number of edges in T without giving away any of the structure of the tree. Lastly, the problem of proving (1) is going to be the most difficult in terms of preserving the zero-knowledge property in the proof system. It will require a more creative approach to construct a zero-knowledge proof system for the MLSTP than what we have considered so far.

3.3.3.3 Coping with Weighted Graphs

So far our work on zero-knowledge proof systems has dealt with only unweighted graphs, i.e. all edge weights are either 0 or 1, corresponding to nonedges and edges respectively. When edge weights are introduced into an interactive proof system, usually the completeness and soundness properties are preserved but the zero-knowledge property is not. When the prover reveals information in the permuted and committed adjacency matrix for the graph, the prover is not only revealing that edges exist but also the weights of the edges. This allows the verifier to discover information about the graph that could not possibly have been computed using a simulator. So far, it does not appear that this issue has been addressed in the literature.

When we consider the decision version of the minimum label spanning tree problem, also known as the bounded label spanning tree problem (BLSTP), the original proof of the NP-completeness of the problem is based on proving that if MLSTP is easily solved then the minimum set covering problem is easily solved (Chang and Leu 1997). Unfortunately, as there is no clear way to convert an arbitrary instance of MLSTP into another known NP-complete problem, we are left with no obvious way of transforming an existing zero-knowledge proof system for the class NP to this problem, as is suggested in the proofs that all languages in NP have zero-knowledge protocols (Goldreich, Micali and Wigderson 1991).

There are several possible options for creating zero-knowledge proof systems for weighted graph problems, however none of these options has been especially fruitful. While some of the options have worked in specific cases, no option has worked in every case and there still remain problems in which no option is feasible (MLSTP). The options that have been considered already are the following:

1. Convert the base problem on weighted graphs to a base problem on unweighted graphs by changing all edge weights that are greater than one to edge weight one.

   In some problems, such as the minimum label spanning tree problem, this option can make the base problem much easier. This can enable a cheater to break the problem instance and impersonate a trusted party. However, this solution seems feasible for the graph partitioning problem.

2. Convert the problem instances so that the solutions use the same number of edges of each edge weight involved and include this number in the common input.

   This is not always a realistic possibility. It can become quite cumbersome to create problem instances in which the solutions are uniform, and it can also make the problem instances much easier for cheaters to break and solve. However, this option appears to work well for the graph coloring problem (considering vertex weights instead of edge weights).

3. Use the reduction from an existing NP-complete problem to the base problem (as is done in a standard proof of NP-completeness) to transform the problem into one that is usable in an existing zero-knowledge proof system.

   Proofs of NP-completeness show two facts. The first is that the base problem is in the class NP. The second fact is that the problem is harder than an existing NP-complete problem, i.e. an instance of the existing NP-complete problem is true if and only if a corresponding instance of the base problem is true. This is most commonly accomplished by transforming an instance of the existing problem into some corresponding instance of the base problem. This leaves us with no way to transform any instance of the base problem into an instance of the existing problem, and hence no way to apply a zero-knowledge proof system for the existing problem to the base problem. However, this approach works well for converting the traveling salesman problem to a sub-graph isomorphism problem (by adding k vertices along edge e with $l(e) = k + 1$ and then searching the new unweighted graph for a cycle of length equal to the length of a minimum TSP tour in the original weighted graph).

Weighted graphs appear to greatly complicate the zero-knowledge proof systems. The three options discussed above clearly are not perfect solutions, but they do seem to work for some particular problems. It is worth considering whether the added complication is worth the trouble. Either the base problem can be converted to an unweighted graph by adding vertices and edges (which increases the number of bits sent between prover and verifier) or the prover is required to send a commitment to an adjacency matrix that is no longer filled with only 0's and 1's (which again increases the number of bits sent). In the first case, the amount of information that needs to be transferred increases, while the problem instances themselves may not be more difficult than instances of a similar base problem on unweighted graphs. In the second case, it becomes much more difficult to satisfy the zero-knowledge property.

4. RESULTS AND DISCUSSION

While it may be one of the best known NP-complete problems, the satisfiability problem is not a practical base problem for a zero-knowledge proof system. First, the amount of information that is required to be computed and transferred in the existing protocol is very large compared the protocols that exist for the other problems discussed in this report. Second, many efficient solvers exist for the problem. For example, the solvers tested during the SAT competition are able to solve instances with millions of variables and millions of clauses. This fact coupled with the data transfer in the zero-knowledge proof system discussed makes the problem very impractical for implementation. Lastly, as of yet there does not exist a method for generating hard instances of the satisfiability problem. Many instances that are known to be difficult were found by a guess-and-check process, which will not be practical for use in a secure protocol. We must be able to create hard instances of whatever base problem is selected.

Graph coloring and equitable coloring are one step closer to being practical base problems for zero-knowledge proof systems than the satisfiability problem. While the probability of catching a cheating prover may not be as high in the protocol for equitable 3-coloring as in some of the protocols using other base problems, we are at least aware of methods for creating difficult problem instances. A difficult problem instance is one in which the existing algorithms are unable to solve optimally in a reasonable amount of time. The set of graphs introduced by The Second DIMACS Implementation Challenge (1992-1993) seems to contain some difficult classes of graph coloring instances. These difficult instances would enable the graph coloring problem to be a good base problem for a zero-knowledge proof system, but a stronger zero-knowledge proof system in which a cheater is more easily discovered must be developed.

Out of the problem classes discussed in this report, the sub-graph isomorphism class appears to be the most promising. In particular, the longest path problem and the sub-graph isomorphism problem itself seem to have the most potential. Currently there do not exist any extremely efficient solvers for the longest path problem, and all sub-graph isomorphism class problems have a zero-knowledge proof system with a probability of catching a cheating prover, taking only 7 rounds to achieve a confidence level of 99%. The protocols are also efficient compared to the existing protocols for other classes of base problems in terms of the amount of data transferred between prover and verifier. Overall, the problems in the sub-graph isomorphism class, with the exception of graph isomorphism, seem to have the most potential.

While the sub-graph isomorphism class is emerging as a useful set of base problems for zero-knowledge proof systems, there is still work to be done. More testing needs to be done on the efficiency of the algorithms for the sub-graph isomorphism problem and its subproblems in order to determine the lower bound on the size of the problem instance for a difficult problem. We must also determine which graph structures are capable of producing the hardest instances in the sub-graph isomorphism class. Is the average instance of the longest path problem harder than

the average case of the minimum bandwidth problem? Which of the problems can we develop difficult instances for in a consistent manner?

Last, but not least, we must consider the latest problems in this area. For example, the minimum label spanning tree, introduced in 1997, could be a promising base problem for a zero-knowledge proof system. However, to be able to utilize this difficult problem, we must first create a valid interactive proof system for the problem that satisfies the zero-knowledge property. The creation of a zero-knowledge proof system involving weighted graphs will allow us to consider many more graph theoretic problems that are currently unusable as base problems.

5. CONCLUSIONS AND FUTURE WORK

Zero-knowledge proof systems have many characteristics that are desirable for determining trustworthy parties in an airborne networking environment. One approach is to base zero-knowledge proof systems on the instances and solutions of NP-complete problem. This report has investigated this approach with a focus on the graph theory problems within the NP-complete and NP-hard classes.

Future research in this area must focus application driven requirements associated with airborne mobile adhoc networks. Protocols used for authentication of user identity, and establishment of mutual trust, cannot constrain either the movement of information or the movement of systems anywhere in the battlespace. Successful implementation of ZKP-based authentication protocols will require that there be a positive impact on both network connectivity and network-user operations. The efficiency and effectiveness of I/A protocols therefore need to be considered against realistic scenarios. MANETs by definition are not static, their configuration change over time; network connections and information routing paths change when nodes are added to, or removed from, the network as new user groups form or nodes are compromised. Mitigating factors such as time-sensitivity of the I/A process, communication channel bandwidth and quality, network dynamics and data flows, user security access requirements, and so on, all need to be accounted for when gauging protocol viability.

6. REFERENCES

Alon, Noga, Raphael Yuster, and Uri Zwick. "Color-Coding." *Journal of the ACM* 42, no. 4 (July 1995): 844-856.

Avanthay, C., A. Hertz, and N. Zufferey. "A Variable Neighborhood Search for Graph Coloring." *European Journal of Operational Research* 151 (2003): 379-388.

Banos, R., C. Gil, J. Ortega, and F.G. Montoya. "Multilevel Heuristic Algorithm for Graph Partitioning." *Lecture Notes in Computer Science* 2611 (2003): 143-153.

Battiti, R., and M. Brunato. "R-Evo: A Reactive Evolutionary Algorithm for the Maximum Clique Problem." Technical Report, Universita Degli Studi di Trento, 2007.

Björklund, A., and T. Husfeldt. "Finding a Path of Superlogarithmic Length." *SIAM Journal of Computing* 32, no. 6 (2003): 1395-1402.

Blum, M. "How to Prove a Theorem So No One Else Can Claim It." *Proceedings of the International Congress of Mathematicians*, 1986: 1444-1451.

Blum, M., P. Feldman, and S. Micali. "Noninteractive Zero Knowledge and Its Applications." *Proceedings of the 20th STOC*, 1988: 103-112.

Brassard, G., and C. Crepeau. "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond." *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, 1986: 188-195.

Brassard, G., D. Chaum, and C. Crepeau. "Minimum Disclosure Proofs." *Jounral of Computer and System Sciences* 37, no. 2 (October 1988): 156-189.

Brélaz, D. "New Methods to Color the Vertices of a Graph." *Communications of the ACM* 22, no. 4 (April 1979): 251-256.

Bui, T.N., and B.R. Moon. "Genetic Algorithm and Graph Partitioning." *IEEE Transactions on Computers* 45, no. 7 (July 1996): 841-855.

Caballero-Gil, P., and C. Hernandez-Goya. "Zero-Knowledge Hierarchical Authentication in Manets." *IEICE Transactions on Information and Systems* E89-D, no. 3 (March 2006): 1288-1289.

Chang, R.S., and S.J. Leu. "The Minimum Labeling Spanning Trees." *Information Processing Letters* 63, no. 5 (1997): 277-282.

Coja-Oghlan, A., and A. Taraz. "Exact and Approximative Algorithms for Coloring G(n,p)." *Random Structures and Algorithms* 24, no. 3 (2004): 259-278.

Consoli, S. "The Development and Application of Metaheuristics for Problems in Graph Theory: A Computational Study." PhD Dissertation, School of Information Systems, Computing and Mathematics, Brunel Universiy, 2008.

Consoli, S., K. Draby-Downman, N. Mladenovic, and J.A.M. Perez. "Greedy Randomized Adaptive Search and Variable Neighbourhood Search for the Minimum Labelling Spanning Tree Problem." *European Journal of Operational Research* 196 (2009): 440-449.

Conte, D., P. Foggia, C. Sansone, and M. Vento. "Thirty Years of Graph Matching in Pattern Recognition." *International Journal of Pattern Recognition and Artificial Intelligence* (World Scientific Publishing Company) 18, no. 3 (2004): 265-298.

Cook, S. "The Complexity of Theorem-Proving Procedures." *Proceeds of the 3rd Annual ACM Symposium on Theory of Computing*, 1971: 151-158.

Cordella, L. P., P. Foggia, C. Sansone, and M. Vento. "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, no. 10 (October 2004): 1367-1372.

Cygan, M., and M. Pilipczuk. "Faster Exact Bandwidth." *Lecture Notes in Computer Science* 5344 (2008): 101-109.

Damgard, I. "Non-Interactive Circuit Based Proofs and Non-Interactive Perfect Zero-Knowledge with Preprocessing." *Proceedings of Eurocrypt*, 1992: 341-355.

De Santo, M., P. Foggia, C. Sansone, and M. Vento. "A Large Database of Graphs and Its Use For Benchmarking Graph Isomorphism Algorithms." *Pattern Recognition Letters* 24 (2003): 1067-1079.

DeSantis, A., G. Di Crescenzo, G. Persiano, and M. Yung. "On Monotone Formula Closure of SZK." *Proc. of the 35th IEEE Symp. on Foundations of Computer Science*, 1994.

DeSantis, A., G. Di Crescenzo, O. Goldreich, and G. Persiano. "The Graph Clustering Problem has a Perfect Zero-Knowledge Interactive Proof." *Information Processing Letters* 69, no. 4 (1999): 201-206.

Desmedt, Y., and Y. Wang. "Efficient Zero-Knowledge Proofs for Some Practical Graph Problems." *Lecture Notes in Computer Science: Security in Communication Networks* 2576 (2003): 290-302.

Diestel, R. *Graph Theory.* Springer, 2006.

Dorn, F. *Planar Sub-graph Isomorphism Revisited.* September 2009. http://arxiv.org/abs/0909.4692v1 (accessed May 2010).

Duff, I.S. "Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)." Technical Report, Research and Technology Division, Boeing Computer Services, Seattle, 1992.

Dwork, C., U. Feige, J. Killian, M. Naor, and M. Safra. "Low Communication 2-Prover Zero-Knowledge Proofs for NP." *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, 1992: 215-227.

Felner, A. "Finding Optimal Solutions to the Graph Partitioning Problem with Heuristic Search." *Annals of Mathematics and Artificial Intelligence* 45, no. 3-4 (December 2005): 293-322.

Foggia, P., C. Sansone, and M. Vento. "A Performance Comparison of Five Algorithms for Graph Isomorphism." *Proceedings of the Third IAPR TC-15 Workshop on Graph Based Representations in Pattern Recognition*, 2001: 188-199.

Foggia, Pasquale. *The VFLib Graph Matching Library, version 2.0.* March 2001. http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html (accessed May 24, 2010).

Fomin, F. V., D. Lokshtanov, V. Raman, B. V.R. Rao, and S. Saurabh. *"Faster Algorithms for Finding and Counting Sub-graphs"*. December 2009. http://arxiv.org/abs/0912.2371v1 (accessed May 2010).

Fortin, S. "The Graph Isomorphism Problem." Technical Report, University of Alberta, 1996.

Galinier, P., A. Hertz, and N. Zufferey. "An Adaptive Memory Algorithm for the k-Coloring Problem." *Discrete Applied Mathematics* 156, no. 2 (2008): 267-279.

Galinier, P., and A. Hertz. "A Survey of Local Search Methods for Graph Coloring." *Computers and Operations Research* 33 (2006): 1547-2562.

Galinier, P., and J.K. Hao. "Hybrid Evolutionary Algorithms for Graph Coloring." *Journal of Combinatorial Optimization* 3 (1999): 379-397.

Garey, M.R., and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W.H. Freeman and Company, 1979.

Goldberg, E., and Y. Novikov. "BerkMin: A Fast and Robust SAT-Solver." *Proc. of Date '02*, 2002: 142-149.

Goldreich, O. "The Graph Clustering Problem has a Perfect Zero-Knowledge Proof." *Theory of Crypto Library*, 1996.

Goldreich, O., and A. Kahan. "How to Construct Constant-Round Zero-Knowledge Proof Systems for NP." *Journal of Cryptology* 9 (1996): 167-189.

Goldreich, O., S. Micali, and A. Wigderson. "Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems." *Journal of the ACM* 38, no. 1 (1991): 691-729.

Golumbic, M.C. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

Grady, L., and E.L. Schwartz. "Isoperimetric Partitioning: A New Algorithm for Graph Partitioning." *SIAM Journal of Scientific Computing* 27, no. 6 (2006): 1844-1866.

Grigoriev, D., and V. Shpilrain. "Zero-Knowledge Authentication Schemes from Actions on Graphs, Groups, or Rings." *CoRR*, 2008.

Grosso, A., M. Locatelli, and W. Pullan. "Simple Ingredients Leading to Very Efficient Heuristics for the Maximum Clique Problem." *Journal of Heuristics* 14 (2008): 587-612.

Helsgaun, K. "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic." *European Journal of Operational Research* 126 (2000): 106-130.

Hernandez-Goya, C., and P. Caballero-Gil. "A New Role of Graph Theory: The Design of Probably Secure Cryptoprotocols." *Information Systems Security*, March/April 2004: 34-43.

Hertz, A., M. Plumettaz, and N. Zufferey. "Variable Space Search for Graph Coloring." *Discrete Applied Mathematics* 156, no. 13 (July 2008): 2551-2560.

Hoos, H.H., and T. Stutzle. "Local Search Algorithms for SAT: An Empirical Evaluation." *Journal of Automated Reasoning* 24, no. 4 (2000): 421-481.

Jia, H. "Hard Instances with Hidden Solutions." PhD Dissertation, University of New Mexico, 2007.

Johnson, D.S., and L.A. McGeoch. "Experimental Analysis of Herustics for the STSP." Chap. 1 in *The Traveling Salesman Problem and its Variations*, edited by Gutin and Punnen, 369-443. Kluwer Academic Publishers, 2002.

Johnson, D.S., and M.A. Trick, . *Volume 26: DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, 1996.

Johnson, D.S., C.R. Aragon, L.A. McGeoch, and C. Schevon. "Optimization by Simultaed Annealing: An Experimental Evaluation, Part I, Graph Partitioning." *Operations Research* 37 (1989): 865-892.

Karger, D., R. Motwani, and G.D.S. Ramkumar. "On Approximating the Longest Path in a Graph." *Algorithmica* (Springer New York) 18, no. 1 (May 1997): 82-98.

Katayama, K., A. Hamamoto, and H. Narihisa. "An Effective Local Search for the Maximum Clique Problem." *Information Processing Letters* 95, no. 5 (September 2005): 503-511.

Kernighan, B.W., and S. Lin. "Partitioning Graphs." *The Bell System Technical Journal*, February 1970: 291-307.

Kim, Y.H., and B.R. Moon. "Lock-Gain Based Graph Partitioning." *Journal of Heuristics* 10 (2004): 37-57.

Knuth, Donald E. "The Stanford GraphBase: A Platform for Combinatorial Algorithms." *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1993: 41-43.

Kurosawa, K., and K. Takai. "A Comment on NIZK for 3-Colorability." *Singapore ICCS/ISITA*, 1992: 274-278.

LeBodic, P., H. Locteau, S. Adam, P. Heroux, Y. Lecourtier, and A. Knippel. "Symbol Detection Using Region Adjacency Graphs and Integer Linear Programming." *10th International Conference on Document Analysis and Recognition*, 2009: 1320-1324.

Lim, A., B. Rodrigues, and F. Xiao. "Heuristics for Matrix Bandwidth Reduction." *European Journal of Operational Research* 174, no. 1 (2006): 69-91.

Lipets, V., N. Vanetik, and E. Gudes. "Subsea: An Efficient Heuristic Algorithm for Sub-graph Isomorphism." *Data Mining and Knowledge Discovery*, May 2009.

Loureiro, R.Z., and A.R.S. Amaral. "An Efficient Approach for Large Scale Graph Partitioning." *Journal of Combinatorial Optimization* 13 (2007): 289-320.

Marinakis, Y., A. Migdalas, and P.M. Pardalos. "A Hybrid Genetic - GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem." *Journal of Combinatorial Optimization* 10 (2005): 311-326.

Marques-Silva, J.P., and K.A. Sakallah. "GRASP: A Search Algorithm for Propositional Satisfiability." *IEEE Transactions on Computers* 48, no. 5 (May 1999): 506-521.

McKay, B.D. "Practical Graph Isomorphism." *Congressus Numerantium* 30 (1981): 45-87.

Pullan, W., and H.H. Hoos. "Dynamic Local search for the Maximum Clique Problem." *Journal of Artificial Intelligence Research* 25 (2006): 159-185.

Simari, G.I. "A Primer on Zero Knowledge Protocols." Technical Report, Universidad Nacional del Sur, 2002.

Simmons, G.J., ed. *Contemporary Cryptology: The Science of Information Integrity.* New York: IEEE, Inc., 1992.

Skiena, S.S. *The Algorithm Design Manual.* London: Springer-Verlag London Limited, 2008.

Solnon, Christine. "AllDifferent-Based Filtering for Sub-graph Isomorphism." *Artificial Intelligence*, 2010: doi:10.1016/j.artint.2010.05.002.

Ullmann, J.R. "An Algorithm for Sub-graph Isomorphism." *Journal of the ACM* 23, no. 1 (January 1976): 31-42.

Vassilevska, V., R. Williams, and S.L.M. Woo. "Confronting Hardness Using a Hybrid Approach." *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Miami: ACM, 2006. 1-10.

Voß, S., and C. Duin. "Look Ahead Features in Metaheuristics." *MIC2003: The Fifth Metaheuristics International Conference*, 2003: 79-1 - 79-7.

Woeginger, G.J. "Exact Algorithms for NP-Hard Problems: A Survey." *Lecture Notes in Computer Science* 2570 (2003): 185-207.

Xiong, Y., B. Golden, and E. Wasil. "A One-Parameter Genetic Algorithm for the Minimum Labeling Spanning Tree Problem." *IEEE Transactions on Evolutionary Computation* 9, no. 1 (2005): 55-60.

Xiong, Y., B. Golden, and E. Wasil. "Worst-Case Behavior of the MVCA Heuristic for the Minimum Labeling Spanning Tree Problem." *Operations Research Letters* 33, no. 1 (2005): 77-80.

Zampelli, S., Y. Deville, and C. Solnon. "Solving Sub-graph Isomorphism Problems with Constraint Programming." *Constraints*, 2010 (to appear).

Zhang, H. "SATO: An Efficient Propositional Prover." *Proc. of the 14th International Conference on Automated Deduction*, 1997: 272-275.

7. LIST OF SYMBOLS AND ABBREVIATIONS

Symbols

$\wedge$	and
$\vee$	or
$\approx$	is isomorphic to
$[k]$	the integers 1 through k
$\neg a$	for Boolean variable a, the complement of a
$S \backslash T$	for sets S and T, $\{s \in S : s \notin T\}$
$E(G)$	the set of edges of graph G
$G = (V, E)$	a graph with vertex set V and edge set E
$\overline{G}$	the complement of graph G
$\|G\|$	the number of vertices in graph G
$\|\|G\|\|$	the number of edges in graph G
$G(n, p)$	the Erdős-Rényi model random graph on n vertices with edge probability p
$G[S]$	the sub-graph of G induced by the set $S \subseteq V(G)$
P_k	a path with k edges
P_k^n	a path P_k with additional edges added between every pair of vertices x, y such that the distance between x and y in the path P_k is at most n
$V(G)$	the set of vertices of graph G
$\Delta(G)$	the maximum degree of the graph G
$\pi\phi$	for permutations π and ϕ, equivalent to $\phi \circ \pi$
$\chi(G)$	the chromatic number of the graph G

Abbreviations

3-SAT	satisfiability problem consisting of clauses with three variables
AN	airborne network
APX	the class of optimization problems with polynomial-time approximation algorithms with approximation ratio bounded by a constant
BLSTP	bounded label spanning tree problem
DIMACS	center for Discrete Mathematics and Theoretical Computer Science
DLS	dynamic local search
E3C	equitable 3-coloring problem
G3C	graph 3-coloring problem
GA	genetic algorithm
GCP	graph clustering problem

GIP	graph isomorphism problem
GNI	graph non-isomorphism problem
GPP	graph partitioning problem
GRASP	greedy randomized adaptive search procedure
HCP	Hamiltonian cycle problem
ISP	independent set problem
KIS	the k-independent set problem
LPP	longest path problem
MANET	mobile ad hoc network
MBP	minimum bandwidth problem
MCP	maximum clique problem
MLSTP	minimum label spanning tree problem
MVCA	maximum vertex covering algorithm of Chang and Leu (1997)
NP	the class of nondeterministic polynomial problems
P	the class of deterministic polynomial problems
QRA	quadratic residuosity assumption
RLS	reactive local search
SA	simulated annealing
SAT	satisfiability problem
SGI	sub-graph isomorphism problem
TSP	traveling salesman problem
VNS	variable neighborhood search
VSS	variable space search
ZKP	zero-knowledge proof system

APPENDIX: ANNOTATED BIBLIOGRAPHY

NP-COMPLETE GRAPH PROBLEMS

Generally considered the standard reference on NP-complete problems:

1. M.R. Garey, D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company: San Francisco, 1979.

GENERAL PAPERS ON ZKP BACKGROUND

2. J. Feigenbaum. "Overview of Interactive Proof Systems and Zero-Knowledge." Contemporary Cryptology: The Science of Information Integrity. (Ed. G. J. Simmons) IEEE Press: 423-439, New York, 1992.

3. O. Goldreich, Y. Oren. "Definitions and Properties of Zero-Knowledge Proof Systems." *Journal of Cryptology*: 1-32, 1994.

4. O. Goldreich. Foundations of Cryptography: Fragments of a Book. Weizmann Institute of Science: 1995.

5. G.J. Simmons, ed. Contemporary Cryptology: The Science of Information Integrity. New York: IEEE, Inc., 1992.

GRAPH ISOMORPHISM

6. G. Brassard, C. Crepeau. "Non-Transitive Transfer of Confidence: A *Perfect* Zero-Knowledge Interactive Protocol for SAT and Beyond." *Proc. of the 27th Annual Symp. on Foundations of Computer Science*: 188-195, 1986.

   Notes: Introduces an idea for a ZKP for graph isomorphism based on the assumption that arbitrarily hard instances of the problem exist. States that the protocol will be formalized in a later paper.

7. D. Conte, P. Foggia, C. Sansone, and M. Vento. "Thirty Years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence* (World Scientific Publishing Company) 18, no. 3(2004): 265-298.

Notes: Discusses many of the graph and sub-graph isomorphism algorithms that existed at the time of publication. Also shows many applications of the problems and algorithms.

8. L. P. Cordella, P. Foggia, C. Sansone, M. Vento. "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10): 1367-1372. Oct. 2004.

Notes: Introduces and describes the VF2 algorithm. Compares VF2 with Nauty and Ullman's algorithm on the graph isomorphism problem with input graphs that are randomly connected, 2D mesh, or bounded valence graphs.

9. P. Foggia, C. Sansone, M. Vento. "A Performance Comparison of Five Algorithms for Graph Isomorphism." *Proc. of the 3rd IAPR TC-15 Workshop on Graph Based Representations in Pattern Recognition*: 188-199, 2001.

Notes: Compares VF2, Nauty, and Ullman's algorithm on benchmark sets of graphs (tested on randomly connected, 2D mesh, and bounded valence graphs). Contains many graphs and plots of the results.

10. P. Foggia. *The VFLib Graph Matching Library, version 2.0.* March 2001. Available at http://amalfi.dis.unina.it/graph/db/vlib.html (accessed May 24, 2010).

Notes: The home of the VF2 algorithm. The C++ code is publicly available at this site.

11. S. Fortin. "The Graph Isomorphism Problem." Technical Report TR 96-20: University of Alberta, July 1996.

Notes: A description of the graph isomorphism problem with a description of some invariants under isomorphism that can be used to reduce the search space. Also discusses Nauty and tests the program with a few specific types of graphs.

12. O. Goldreich. Foundations of Cryptography: Fragments of a Book. Weizmann Institute of Science: 1995.

Notes: Presents a perfect zero-knowledge proof for the graph isomorphism problem. Goes through a formal and thorough proof that the protocol presented is a zero-knowledge proof system using simulators.

13. O. Goldreich, S. Micali, A. Wigderson. "Proofs that Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems." *Journal of the ACM* 38(1): 691-729, 1991.

Notes: Presents a perfect zero-knowledge proof system for the graph isomorphism problem. Contains a thorough discussion and proof that the protocol is a perfect zero-knowledge proof system. Discusses a modification to the protocol to enable parallel execution instead of sequential.

14. D. Grigoriev, V. Shpilrain. "Zero-Knowledge Authentication Schemes from Actions on Graphs, Groups, or Rings." *CoRR*: 2008.

Notes: Discusses the problem as a *promise* problem, i.e. find a particular isomorphism between the two graphs (not just any isomorphism). Outlines the protocol and proves that successful forgery in the protocol is equivalent to solving the instance of the graph isomorphism problem.

15. C. Hernandez-Goya, P. Caballero-Gil. "A New Role of Graph Theory: The Design of Probably Secure Cryptoprotocols." *Information Systems Security*: 34-43, March/April 2004.

Notes: Mentions methods for creating difficult instances of the graph isomorphism problem. Discusses several (non-zero-knowledge) protocols for the graph isomorphism problem, and then improves upon the ideas to create a general zero-knowledge proof system for any graph problem.

16. B.D. McKay. "Practical Graph Isomorphism." *Congressus Numerantium* 30 (1981): 45-87.

Notes: Introduces and describes the mathematical methods behind the Nauty algorithm.

17. R. Mun (Advisor: R. Williams). "15-453 FLAC: Graph Isomorphism."

Notes: Presents some background and explanation of the graph isomorphism problem. Discusses some possible approaches to finding an efficient algorithm for the problem and mentions both the positive and negative sides to each approach.

18. J. Pieprzyk, T. Hardjono, J. Seberry. "Zero Knowledge Proof Systems." From Fundamentals of Computer Security. Springer-Verlag: 409-431, 2003.

Notes: Defines and introduces zero-knowledge proof systems. Presents an interactive proof system for the graph isomorphism problem and proves that it is a zero-knowledge proof system in a thorough manner.

19. J. Rothe. Complexity Theory and Cryptology. Springer-Verlag: 386-393, 2005.

Notes: Discusses and outlines a zero-knowledge proof system for the graph isomorphism problem, along with a discussion of the protocol. Discusses various possible commitment schemes to use in the protocol.

20. G. I. Simari. "A Primer on Zero Knowledge Protocols." *Universidad Nacional del Sur*: June 27, 2002.

Notes: Introduces and defines the properties of zero-knowledge proof systems. Outlines a zero knowledge protocol for the problem through a specific example. Discusses why the protocol is zero-knowledge and demonstrates a forgery algorithm.

GRAPH NON-ISOMORPHISM

21. O. Goldreich. Foundations of Cryptography: Fragments of a Book. Weizmann Institute of Science: 1995.

Notes: Discusses an interactive proof system for GNI, and then goes on discuss how to modify the protocol to achieve a zero-knowledge proof system in a later section.

22. O. Goldreich, S. Micali, A. Wigderson. "Proofs that Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems." *Journal of the ACM* 38(1): 691-729, 1991.

Notes: Presents an interactive proof system for graph non-isomorphism and then extends this proof system to a perfect zero-knowledge proof system for the problem. Mentions that the protocol discussed can also be run in parallel instead of sequentially.

23. J. Pieprzyk, T. Hardjono, J. Seberry. "Zero Knowledge Proof Systems." From Fundamentals of Computer Security. Springer-Verlag: 409-431, 2003.

Notes: Presents an interactive proof system for the graph non-isomorphism problem. States that the protocol is zero-knowledge (with reference to a proof). Discusses the relationship between this protocol and the one for the graph isomorphism problem.

SUB-GRAPH ISOMORPHISM

24. N. Alon, Raphael Yuster, and Uri Zwick. "Color-Coding." *Journal of the ACM* 42, no. 4 (July 1995): 844-856.

Notes: Introduces the color-coding method of solving the sub-graph isomorphism problem. This algorithm solves certain subcases of the SGI in polynomial time using the concept of treewidth and tree decompositions.

25. L. P. Cordella, P. Foggia, C. Sansone, M. Vento. "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10): 1367-1372. Oct. 2004.

Notes: Introduces and describes the VF2 algorithm. Compares VF2 to Ullman's algorithm, and concludes that VF2 performs better, especially when sub-graph has more than 20 nodes. Tests the algorithm on graphs derived from large line drawings of publicly available images.

26. F. Dorn. "Planar Sub-graph Isomorphism Revisited." September 2009. Available on the arXiv: http://arxiv.org/abs/0909.4692v1 (accessed May 2010).

Notes: Theoretical results on planar sub-graph isomorphism.

27. P. Foggia. *The VFLib Graph Matching Library, version 2.0.* March 2001. Available at http://amalfi.dis.unina.it/graph/db/vlib.html (accessed May 24, 2010).

Notes: The home of the VF2 algorithm. The C++ code is publicly available at this site.

28. F.V. Fomin, D. Lokshtanov, V. Raman, B.V.R. Rao, and S. Saurabh. "Faster Algorithms for Finding and Counting Sub-graphs." December 2009. Available on the arXiv: http://arxiv/org/abs/0912.2371v1 (accessed May 2010).

Notes: Theoretical results on the sub-graph isomorphism problem using the notion of treewidth and randomized algorithms.

29. D. Grigoriev, V. Shpilrain. "Zero-Knowledge Authentication Schemes from Actions on Graphs, Groups, or Rings." *CoRR*: 2008.

Notes: Describes the usual sub-graph isomorphism protocol, but commits to the graph by embedding it in a larger graph. Contains a few short notes about the protocol.

30. P. LeBodic, H. Locteau, S. Adam, P. Heroux, Y. Lecourtier, and A. Knippel. "Symbol Detection Using Region Adjacency Graphs and Integer Linear Programming." *10th International Conference on Documnt Analysis and Recognition*, 2009: 1320-1324.

Notes: Formulates the sub-graph isomorphism problem as an integer linear program. Contains experimental results that are specific to architectural applications.

31. V. Lipets, N. Vanetik, E. Gudes. "Subsea: an efficient heuristic algorithm for sub-graph isomorphism." *Data Mining and Knowledge Discovery*: 2009.

Notes: Introduces a new algorithm that finds all sub-graphs in the second graph that are isomorphic to the first (instead of just one like VF2 or Ullman's). Performs well when all sub-graphs are desired, but when just one is needed, it is often outperformed.

32. H. Shang, Y. Zhang, X. Lin, J. X. Yu. "Taming Verification Hardness: An Efficient Algorithm for Testing Sub-graph Isomorphism." *Proc. of the VLDB Endowment*, 1(1): 364-374, Aug. 2008.

Notes: Introduces and describes the QuickSI algorithm for testing sub-graph isomorphism. Compares and evaluates QuickSI against the Ullman algorithm. Both algorithms use branch and bound, but QuickSI encodes an ordering while Ullman is a random ordering.

33. C. Solnon. "AllDifferent-Based Filtering for Sub-graph Isomorphism." *Artificial Intelligence*, 2010: doe: 10.1016/j.artint.2010.05.002.

Notes: Introduces a new filtering algorithm for the sub-graph isomorphism and shows experimental results comparing the new algorithm with VF2 and other existing algorithms.

34. A. Takura. "Automated Generation of Communications Software from Service Specifications Described by State Transition Rules." *Proc. of the Thirtieth Hawaii International Conference*: 472-480, Jan. 1994

Notes: Develops a procedure for automated generation of software for telephone service based on the sub-graph isomorphism problem. The graphs that are developed for this purpose can find sub-graph isomorphisms at a "practical speed".

35. J.R. Ullman. "An Algorithm for Sub-graph Isomorphism." *J. Assoc. for Computing Machinery*, 23: 31-42, 1976.

Notes: Introduces Ullman's algorithm for the graph isomorphism and sub-graph isomorphism problem. Tests the algorithm on several different types of graphs.

36. S. Zampelli, Y. Deville, and C. Solnon. "Solving Sub-graph Isomorphism Problems with Constraint Programming." *Constraints*, 2010 (to appear).

Notes: Introduces a new filtering algorithm and tests the new algorithm against other algorithms such as VF2.

HAMILTONICITY

37. M. Blum. "How to Prove a Theorem So No One Else Can Claim It." *Proceedings of the International Congress of Mathematicians*: 1444-1451, 1986.

Notes: Outlines and discusses a zero-knowledge proof system for the Hamiltonian cycle problem. Presents the protocol using locked boxes instead of encryption/commitments. Proves that the properties of a zero-knowledge proof system are satisfied in the given protocol.

38. P. Caballero-Gil, C. Hernandez-Goya. "Zero-Knowledge Hierarchical Authentication in MANETs." *IECE Trans. Inf. And Syst.*, Vol. E89-D, No. 3: 1288-1289, March 2006.

Notes: Uses a zero-knowledge proof system for the Hamiltonian cycle problem to implement a hierarchical scheme. Notes that zero-knowledge proof system involved need not be based on the Hamiltonian cycle problem and that any other hard graph problem would suffice.

39. B. Chazelle. "The security of knowing nothing." Nature, vol. 446: 992-993, April 2007.

Notes: Begins with a basic introduction to zero-knowledge proof systems and discusses a real-world application of the Hamiltonian cycle problem. Discusses how to hide the private information instead of revealing all.

40. I. Damgård. "Non-Interactive Circuit Based Proofs and Non-Interactive Perfect Zero-Knowledge with Preprocessing." *Proc. of Eurocrypt*: 341-355, 1992.

Notes: Presents an interactive argument with preprocessing from a paper in the references. The author goes on to critique this model and present a more efficient preprocessing and proof phase based on an assumption that collision free hash functions exist.

41. D. Grigoriev, V. Shpilrain. "Zero-Knowledge Authentication Schemes from Actions on Graphs, Groups, or Rings." *CoRR*: 2008.

Notes: Observes that this problem is a special case of the sub-graph isomorphism problem, which is then discussed (see sub-graph isomorphism).

42. G. Gutin, D. Karapetyan. "16: Greedy Like Algorithms for the Traveling Salesman and Multidimensional Assignment Problems." Advances in Greedy Algorithms. W. Bednorz (Ed.), I-Tech, Vienna, Austria: 291-304, Nov. 2008.

Notes: Introduces the asymmetric TSP and the symmetric TSP problems along with greedy algorithms for both. Introduces the greedy algorithm, NN algorithm, and Patch algorithm, and tests them on some instances from TSPLIB.

43. R. Hassin, A. Keinan. "Greedy Heuristics with Regret, with Application to the Cheapest Insertion Algorithm for the TSP." *Operations Research Letters*, 36: 243-246, 2008.

Notes: Introduces a greedy algorithm with partial regret (reconsider past decisions). Compares the standard algorithm with this algorithm allowing regret, and tests the algorithm on the TSPLIB instances, showing a reduction in average error.

44. K. Helsgaun. "An Effective Implementation of the Lin-Kernighan Traveling Salesman Hueristic." *European Journal of Operational Research*, 126 (2000): 106-130.

Notes: Introduces an implementation of the symmetric TSP and finds optimal solutions of real-world instances.

45. D.S. Johnson, L.A. McGeoch. "Chapter 1: Experimental Analysis of Heuristics for the STSP." The Traveling Salesman Problem and its Variations. Gutin, Punnen (Eds.), Kluwer Academic Publishers: 369-443, 2002.

Notes: Discusses and compares the relevant algorithms (from 2002). Testing is done on several random instances as well as instances from TSPLIB. Only instances with more than 1000 nodes were considered, as ones with fewer than 1000 nodes are generally considered too easy.

46. D. Kaur, M.M. Murugappan. "Performance Enhancement in Solving Traveling Salesman Problem using Hybrid Genetic Algorithm." *Proc. of the IEEE NAFIPS Conference*: May, 2008.

Notes: Introduces a hybrid genetic algorithm for TSP and compares it to NN (nearest neighbor) and pure GA (genetic algorithm).

47. D. Lapidot, A. Shamir. "A one-round, two-prover, zero-knowledge protocol for NP." *Combinatorica*, 15(2): 203-214, June, 1995.

Notes: Quickly describes a basic zero-knowledge proof system for the Hamiltonian cycle problem with one prover. Extends this basic protocol to a zero-knowledge proof system for the Hamiltonian cycle problem with two provers and one verifier. Thoroughly proves that the properties of a zero-knowledge proof system are satisfied.

48. D. Lin, X. Wu, D. Wang. "Exact Heuristic Algorithm for Traveling Salesman Problem." *Proc. of 9th International Conference for Young Computer Scientists*: 9-13, Nov. 2008.

Notes: Introduces a new heuristic algorithm (BACHA) based on branch-and-cut for the TSP. Tests BACHA against the normal GA on some benchmark instances from TSPLIB95. Also compares it with a version of the Lin-Kernighan algorithm.

49. Y. Marinakis, A. Migdalas, P.M. Pardalos. "A Hybrid Genetic – GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem." *Journal of Combinatorial Optimization*, 10: 311-326, 2005.

Notes: Discusses the different types of algorithms available for TSP as of 2005. Contains a list ranking all of the best known algorithms by average quality. Introduces a new algorithm that is tested on instances from TSPLIB and compares it with existing algorithms.

50. M. Nguyen, S. Vadhan. "Zero Knowledge with Efficient Provers." *Proc. of the 38th Annual Symposium on Theory of Computing*: 287-295, 2006.

Notes: Discusses the idea of 1-out-of-2 binding commitments mostly for the "entropy approximation" problem, but with references to graph isomorphism, 3-colorability, and hamiltonicity. Outlines the usual Hamiltonian cycle zero-knowledge proof system.

51. J.W. Pepper, B.L. Golden, E.A. Wasil. "Solving the Traveling Salesman Problem with Annealing-Based Heuristics: A Computational Study." *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 32(1): 72-77, Jan. 2002.

Notes: Compares several annealing-based heuristics for TSP by testing on instances from TSPLIB. Algorithms compared: SA (simulated annealing), TA (threshold accepting), RRT (record-to-record travel), and DA (demon algorithm).

52. G. Reinelt. "TSPLIB – A Traveling Salesman Problem Library." *ORSA Journal on Computing*, 3(4): 376-384, 1991.

Notes: Discusses the contents of the set of benchmark instances for the traveling salesman problem: TSPLIB.

53. S.S. Skiena. The Algorithm Design Manual. Springer-Verlag: London, 2008.

Notes: Introduces both the Hamiltonian cycle problem and the traveling salesman problem with general background information as well as a short discussion on efficient algorithms.

54. F. Zhao, J. Dong, S. Li, J. Sun. "An Improved Ant Colony Optimization Algorithm with Embedded Genetic Algorithm for the Traveling Salesman Problem." *Proc. of the 7th World Congress on Intelligent Control and Automation*: 7902-7906, June 2008.

Notes: Introduces a new ACO (ant colony optimization) algorithm and tests it against existing ACO algorithms. Testing is done on benchmark instances from TSPLIB.

LONGEST PATH PROBLEM

55. A. Björklund, T. Husfeldt. "Finding a Path of Superlogarithmic Length." *SIAM Journal of Computing*, 32(6): 1395-1402, 2003.

Notes: Introduces a polynomial-time algorithm to find a long (defined in paper) path in a graph. Does not contain experimental results for the algorithm, but contains a more theoretical analysis of the problem and algorithm.

56. D. Karger, R. Motwani, and G.D.S. Ramkumar. "On Approximating the Longest Path in a Graph." *Algorithmica* (Springer New York) 18, no. 1 (May 1997): 82-98.

Notes: Considers several different (inexact) longest path algorithms and compares their performance. Also includes hardness results of the problem.

57. S.N.N. Pandit. "Some Observations on the Longest Path Problem." *Operations Research*, 12(2): 361-364, 1964.

Notes: Critiques a paper that discusses the traveling salesman problem by Hardgrave and Nemhauser where the problem is transformed into the longest path problem. Concludes that this transformation is unhelpful, as the longest path problem is also a difficult problem.

58. V. Vassilevska, R. Williams, S.L.M. Woo. "Confronting Hardness Using a Hybrid Approach." Technical Report, Computer Science Department, Carnegie Mellon University: April 2005.

Notes: Introduces an algorithm that will either find the exact solution in sub-exponential time or approximate the solution in polynomial time. Compares this to the runtimes of other algorithms for the longest path problem.

59. S. Voß and C. Duin. "Look Ahead Features in Metaheuristics." *MIC2003: The Fifth Metaheuristics International Conference,* 2003: 79-1 – 79-7.

Notes: Introduces the Pilot algorithm and shows applications to TSP.

MINIMUM BANDWIDTH PROBLEM

60. M. Cygan, M. Pilipczuk. "Faster Exact Bandwidth." *Lecture Notes in Computer Science*, 5344: 101-109, 2008.

Notes: Discusses exact algorithms for the minimum bandwidth problem. Introduces a new algorithm that has time-complexity $O^*(5^n)$. Also includes some background information on the problem itself.

61. I.S. Duff. "Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)." Technical Report TR/PA/92/86, Research and Technology Division, Boeing Computer Services, Seattle, WA: Oct. 1992.

Notes: Describes the Harwell-Boeing Sparse Matrix Collection that is available online. Gives the background for each class of instances along with some detail on what type of instance the class contains.

62. A. Lim, B. Rodrigues, F. Xiao. "Heuristics for Matrix Bandwidth Reduction." *European Journal of Operational Research*, 174(1): 69-91, 2006.

Notes: Introduces new heuristic algorithms for the minimum bandwidth problem and tests them against the GPS algorithm, GRASP, tabu search, etc., using the Harwell-Boeing Sparse Matrix Collection.

63. V. Vassilevska, R. Williams, S.L.M. Woo. "Confronting Hardness Using a Hybrid Approach." Technical Report, Computer Science Department, Carnegie Mellon University: April 2005.

Notes: Introduces a hybrid algorithm for the minimum bandwidth problem. Offers an algorithm that either solves the problem exactly or approximates it in polynomial time. Discusses the approximation ratio achieved if an exact solution cannot be found.

64. G.J. Woeginger. "Exact Algorithms for NP-Hard Problems: A Survey." *Lecture Notes in Computer Science*, 2570: 185-207, 2003.

Notes: Outlines and discusses the algorithm of Feige and Kilian with time-complexity $O^*(20^n)$ that is based on the technique of pruning the search tree. Concludes that it is still an open problem as to whether the problem has an exact algorithm with time-complexity $O^*(2^n)$.

GRAPH CLUSTERING

65. A. De Santis, G. Di Crescenzo, O. Goldreich, G. Persiano. "The Graph Clustering Problem has a Perfect Zero-Knowledge Interactive Proof." *Information Processing Letters* 69(4): 201-206, 1999.

Notes: Develops a perfect zero-knowledge interactive proof system for the problem based on four subprotocols: the first two show that the input graphs fall into exactly *c* clusters, the last two show that the cluster sizes are exactly the positive integers specified.

66. O. Goldreich. "The Graph Clustering Problem has a Perfect Zero-Knowledge Proof." *Theory of Crypto Library*: 1996.

Notes: Presents a proof (no protocol) based on the threshold formula that a perfect zero-knowledge proof exists for the problem. The proof is valid for inputs with at most 5 graphs.

GRAPH K-COLORABILITY

67. H. Al-Omari, K.E. Sabri. "New Graph Coloring Algorithms." *American Journal of Mathematics and Statistics*, 2(4): 439-441, 2006.

Notes: Proposes and discusses two new heuristic graph coloring algorithms. Compares them against basic existing algorithms such as first fit and concludes that the new heuristic algorithms perform better.

68. C. Avanthay, A. Hertz, N. Zufferey. "A Variable Neighborhood Search for Graph Coloring." *European Journal of Operational Research*, 151: 379-388, 2003.

Notes: Introduces and discusses the variable neighborhood search (VNS) algorithm for the graph coloring problem. Compares VNS with Tabucol and GH (genetic hybrid) using the DIMACS benchmark graphs.

69. I. Blochliger, N. Zufferey. "A Graph Coloring Heuristic Using Partial Solutions and a Reactive Tabu Scheme." *Computers and Operations Research*, 35: 960-975, 2008.

Notes: Introduces and discusses four new algorithms based on PartialCol. Tested against and compared to GH and MMT algorithms on some well-known benchmark graphs. Compares the best colorings found by each algorithm.

70. M. Blum. "How to Prove a Theorem So No One Else Can Claim It." *Proceedings of the International Congress of Mathematicians*: 1444-1451, 1986.

Notes: Outlines a zero-knowledge proof system for the graph 3-colorability problem using locked boxes. Discusses briefly Goldreich, Micali, and Wigderson's (1991) zero-knowledge proof system for 3-colorability.

71. M. Blum, P. Feldman, S. Micali. "Noninteractive Zero Knowledge and its Applications." *Proc. of 20th STOC*: 103-112, 1988.

Notes: Outlines a noninteractive zero-knowledge proof of 3-colorability based on the quadratic residue assumption (number theory). Discusses the limitations of the single-theorem protocol, and improves the protocol to a more general noninteractive zero-knowledge proof for 4-colorability.

72. D. Brelaz. "New Methods to Color the Vertices of a Graph." *Communications of the ACM*, 22(4): 251-256, April 1979.

Notes: Introduces and discusses the DSATUR algorithm. Contains a comparison and testing of three versions of DSATUR against the algorithms that were current at the time.

73. A. Coja-Oghlan, A. Taraz. "Exact and Approximative Algorithms for Coloring *G(n, p)*." *Random Structures and Algorithms*, 24(3): 259-278, 2004.

Notes: Discusses optimal and approximative coloring algorithms for random graphs *G(n, p)*. Presents polynomial-time optimal algorithms for specific ranges of values for *p*.

74. I. Devarenne, A. Caminada, H. Mabed. "Analysis of Adaptive Local Search for Graph Coloring Problem." *The 6th Metaheuristics International Conference*: 1204-1 – 1204-6, 2005.

Notes: Introduces and discusses a new local search method M/L/D/C. Compares M/L/D/C with the mutation and selection algorithm and the tabu search algorithm (both by Dorne and Hao). Analysis and comparison done on CNET instances of the graph coloring problem.

75. I. Devarenne, H. Mabed, A. Caminada. "Optimization by Extension-Restriction Neighborhood in Local Search: Application to Graph Coloring Problem." *Proc. of 20th European Simulation and Modeling Conference*: Oct. 2006.

Notes: Presents a new local search algorithm and compares it with other algorithms that implement either partial neighborhood exploration (local search) or total neighborhood exploration (Tabu search). Evaluates the new algorithm on the DIMACS graphs and compares it with DSATUR, AMACOL, and several tabu methods.

76. I.M. Diaz, P. Zabala. "A Branch-and-Cut Algorithm for Graph Coloring." *Proc. of the Computational Symp. on Graph Coloring and its Generalization*: 2002.

Notes: Introduces and discusses a branch-and-cut algorithm based on integer linear programming. Compares the new algorithm with DSATUR. Concludes that the new algorithm was able to solve more instances of the graph coloring problem than DSATUR given a 2-hour time limit.

77. R. Dorne, J.K. Hao. "A New Genetic Local Search Algorithm for Graph Coloring." *Lecture Notes in Computer Science*, 1498: 745-754, 1998.

Notes: Presents a new genetic local search algorithm based on independent sets and Tabu search. Tests and compares the new algorithm on DIMACS benchmarks against XRLF, EDM, and Fleurent and Ferland's algorithm. While the algorithm is slower, it is able to find better colorings the other algorithms in the graphs that are analyzed.

78. R. Dorne, J.K. Hao. "3: Tabu Search for Graph Coloring, T-Coloring and Set T-Colorings." Metaheuristics '98: Theory and Applications. Kluwer Academic Publishers: 33-48, 1998.

Notes: Introduces the algorithm GTS (Generic Tabu Search). Compares the algorithm against Fleurent and Ferland, EDM, and XRLF. Shows mixed results on the performance of the algorithm.

79. P. Galinier, J.K. Hao. "Hybrid Evolutionary Algorithms for Graph Coloring." *Journal of Combinatorial Optimization*, 3: 379-397, 1999.

Notes: Introduces and discusses the hybrid evolutionary algorithms (HEAs) 'HCA' for the graph coloring problem. Tests and compares HCA to the Tabu search algorithm, against which HCA is shown to outperform in both power and speed.

80. P. Galinier, A. Hertz. "A survey of local search methods for graph coloring." *Computers and Operations Research*, 33: 1547-2562, 2006.

Notes: Discusses several graph coloring algorithms, including Tabucol, which (even though over 20 years old) is still frequently used either alone or as part of a hybrid algorithm. Highlights the differences between different algorithm methods.

81. P. Galinier, A. Hertz, N. Zufferey. "An Adaptive Memory Algorithm for the *k*-Coloring Problem." *Discrete Applied Mathematics*, 156(2): 267-279, 2008.

Notes: Introduces, describes, and discusses Amacol. Compares and tests Amacol against Tabucol, GH, DSATUR, Long_TABU, and Short_TABU on the DIMACS graphs. Concludes that Amacol is competitive with the existing algorithms.

82. O. Goldreich. Foundations of Cryptography: Fragments of a Book. Weizmann Institute of Science: 1995.

Notes: Presents the same protocol as most other references with a thorough description and proof that the protocol is a zero-knowledge proof system. Also contains a discussion in the section's concluding remarks on constant round and efficient protocols for G3C, as well as explicitly constructs a round efficient zero-knowledge proof system for G3C.

83. O. Goldreich, A. Kahan. "How to Construct Constant-Round Zero-Knowledge Proof Systems for NP." *Journal of Cryptology,* 9: 167-189, 1996.

Notes: Describes and outlines an efficient (using a constant number of rounds) zero-knowledge proof system for the graph colorability problem and proves that it satisfies the necessary properties.

84. O. Goldreich, S. Micali, A. Wigderson. "Proofs that Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems." *Journal of the ACM* 38(1): 691-729, 1991.

Notes: Outlines two protocols: one using locked boxes and keys (for understanding), the other using a digital implementation (for practical use). Contains a thorough proof that the protocol is in fact a zero-knowledge interactive proof, as well as a discussion of how to construct constant-round zero-knowledge proof systems for the problem.

85. D. Grigoriev, V. Shpilrain. "Zero-Knowledge Authentication Schemes from Actions on Graphs, Groups, or Rings."

Notes: Outlines a protocol for *k*-colorability that is based on sending a commitment to the coloring through an isomorphic copy of the graph, since colorability is preserved under isomorphism.

86. A. Hertz, M. Plumettaz, N. Zufferey. "Variable Space Search for Graph Coloring." *Discrete Applied Mathematics,* 156(13): 2551-2560, July 2008.

Notes: Introduces the variable space search (VSS) algorithm as an extension of the variable neighborhood search (VNS) algorithm. Runs tests on some graphs from the DIMACS challenge, and compares VSS with TabuCol, PartialCol, GH, MOR and MMT algorithms.

87. H.H. Hoos, T. Stutzle. "Local Search Algorithms for SAT: An Empirical Evaluation." *Journal of Automated Reasoning*, 24(4): 421-481, 2000.

Notes: Contains some algorithmic discussion of the graph colorability problem through transformations from the satisfiability problem. Also contains results from testing done on solving graph coloring instances by GSAT, Novelty, and WalkSAT.

88. D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon. "Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning." *Operations Research*, 39(3): 378-406, 1991.

Notes: Presents a few different algorithms for graph coloring based on simulated annealing. Most of the algorithms require large amounts of time to produce colorings that are close to optimal. Includes experimental data on random graphs of 1000 nodes.

89. W. Klotz. "Graph Coloring Algorithms." 2002. Available at: http://www.math.tu-clausthal.de/Arbeitsgruppen/Diskrete-Optimierung/publications/2002/gca.pdf

Notes: Discusses several different existing algorithms, such as RLF and DSATUR, and introduces a new algorithm based on a heuristic called BSC: Backtracking Sequential Coloring. Compares the runtimes and quality of solution for these algorithms by testing on random graphs on 60 vertices with different edge-densities.

90. S. G. Krantz. "Zero Knowledge Proofs." *Mathematical Adventures for Students and Amateurs*, Spectrum Series, MAA: Washington, D.C., 2006.

Notes: A very basic discussion of 4-colorability with a focus on the encryption methods used. Contains an introduction to RSA encryption and the idea of zero-knowledge proof systems.

91. K. Kurosawa, K. Takai. "A Comment on NIZK for 3-Colorability." *Singapore ICCS/ISITA*: 274-278, 1992.

Notes: Outlines and proves a more efficient noninteractive zero knowledge proof system for 3-colorability based on M. Blum's "Noninteractive Zero Knowledge and its Applications." Protocol is very similar to Blum's with slight modifications.

92. A. Lim, Y. Zhu, Q. Lou, B. Rodrigues. "Heuristic Methods for Graph Coloring Problems." *ACM Symp. on Applied Computing*: 933-939, 2005.

Notes: Introduces a new algorithm based on Tabu search combined with an optimizer to fix priorities. Tests and compares the algorithm against some basic algorithms. Testing is done on benchmark geometric graphs from COLORING '02.

93. D.W. Matula, L.L. Beck. "Smallest-Last Ordering and Clustering and Graph Coloring Algorithms." *Journal of the Association for Computing Machinery*, 30(3): 417-427, July 1983.

Notes: Goes over the basic ideas of smallest-last ordering for a greedy algorithm. Presents upper bounds on the number of colors required by the algorithm for specific classes of graphs.

94. J. Pieprzyk, T. Hardjono, J. Seberry. "Zero Knowledge Proof Systems." From Fundamentals of Computer Security. Springer-Verlag: 409-431, 2003.

Notes: Outlines an interactive proof for 3-colorability, which relies on the assumption that there is a secure probabilistic encryption, as well as a proof that the protocol is computational zero knowledge. Discusses why either encryption or a bit commitment scheme is a necessary ingredient in a computationally zero-knowledge proof system.

95. J. Rothe. "Heuristics versus Completeness for Graph Coloring." *Chicago Journal of Theoretical Computer Science*, 2000(1): 1-16, 2000.

Notes: Studies the complexity of the graph coloring problem when considering input graphs that can be solved by a given heuristic *A* (A-G3C). All heuristics are based on sequential algorithms. Proves that A-G3C is NP-complete for the algorithms considered.

96. G. I. Simari. "A Primer on Zero Knowledge Protocols." *Universidad Nacional del Sur*: June 27, 2002.

Notes: Introduces and defines the concept of zero-knowledge proof systems. Outlines the standard zero-knowledge proof system in which the verifier selects one edge at random. Discusses why this is a zero-knowledge proof system.

97. T. Stutzle. "Introduction to Stochastic Local Search." Presentation given at *ANTS* 2006. Available at: http://iridia.ulb.ac.be/ants2006/tutorial_slides/stuetzle_tutorial_slides.pdf

Notes: Gives an overview of stochastic search methods such as simulated annealing, Tabu search, dynamic local search, and iterative local search (and others). Discusses Tabu search for the graph coloring problem.

98. C. R. Subramanian, M. Furer, C. E. Madhavan. "Algorithms for Coloring Semi-Random Graphs." : 125-158, 1998.

Notes: Shows the existence of and describes polynomial-time algorithms that almost surely succeed in coloring semi-random graphs $G_{SB}(n, p, k)$ for certain ranges of values for p, i.e. a graph supplied by an opponent that will add each edge with probability p or 1-p.

EQUITABLE COLORING

99. H. Furmańczyk, M. Kubale. "The Complexity of Equitable Vertex Coloring of Graphs." *Journal of Applied Computer Science*, 13(2): 95-107, 2005.

Notes: Introduces the equitable coloring problem and discusses the complexity of the problem. Lists which graphs can be equitably colored in polynomial time. Also introduces two polynomial-time heuristic algorithms based on a greedy method.

100. W. Meyer. "Equitable Coloring." *The American Mathematical Monthly*, 80(8): 920-922, Oct. 1973.

Notes: Introduces the notion of equitable coloring and proves some basic results on equitable coloring numbers of graphs.

101. "Equitable Coloring." *Wikipedia: The Free Encyclopedia*. Available at: http://en.wikipedia.org/wiki/Equitable_coloring

Notes: Contains a good introduction to the concept of equitable coloring with examples. Discusses the NP-completeness of the problem and applications.

GRAPH K-EDGE-COLORABILITY

102. R. Venkatesan, L. Levin. "Random Instances of a Graph Coloring Problem are Hard." *Proc. of the Annual ACM Symposium on Theory of Computing*: 217-222, 1988.

Notes: Looks at the graph edge-coloring problem on random digraphs by its inversion problem: color the edges of the graph so as to obtain the specified coloring. Proves that these problems are hard on average by reducing it to the random tiling problem (RTP) and forcing the solver to transform the RTP solution into an edge-coloring of the graph.

INDEPENDENT SETS/MAXIMUM CLIQUE

103. R. Battiti, M. Brunato. "R-Evo: A Reactive Evolutionary Algorithm for the Maximum Clique Problem." Universita Degli Studi di Trento, Technical Report DIT-07-034: May 31, 2007.

Notes: Introduces R-Evo and RLS-Evo and compares them with EA/G and RLS for the maximum clique problem. Also discusses model-based algorithms. Contains data comparing runtime and performance of EA/G against R-Evo on the DIMACS graphs.

104. I. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo. "The Maximum Clique Problem." Handbook of Combinatorial Optimization. Kluwer Academic Publishers, D.Z. Du and P.M. Pardalos (Eds.): 1999.

Notes: An overview of the different types of heuristic algorithms for solving the maximum clique problem. Tests the algorithms on the DIMACS graphs.

105. P. Caballero-Gil. "Zero-Knowledge Proof for the Independent Set Problem." *IEICE Trans. on Fund. of Electronics, Communications, and Computer Science*: 1301-1302, May 2005.

Notes: Presents a zero-knowledge proof system for the independent set problem with a commitment scheme based on the discrete log problem. Discusses the efficiency/complexity of the protocol.

106. L. Cavique, C. Rego, I. Themido. "A Scatter Search Algorithm for the Maximum Clique Problem." Essays and Surveys in Metaheuristics. Kluwer Academic Publishers: 227-244, 2001.

Notes: Introduces a new algorithm based on scatter search (to explore new regions), and tabu search (to improve the new solutions found). Tests the SS algorithm on the DIMACS graphs against several different tabu search algorithms.

107. Y. Desmedt, Y. Wang. "Efficient Zero-Knowledge Proofs for Some Practical Graph Problems." *Lecture Notes in Computer Science: Security in Communication Networks*, 2576: 290-302, 2003.

Notes: Discusses the independent set problem as a special case of the *k*-independent set problem. Outlines and proves zero knowledge protocols for both of these problems, and then exhibits a transformation from the *k*-independent set problem to the independent set problem.

108. C. Friden, A. Hertz, D. de Werra. "Tabaris: An Exact Algorithm Based on Tabu Search for Finding a Maximum Independent Set in a Graph." *Computers Opns. Res.*, 17(5): 437-445, 1990.

Notes: Describes, outlines, and discusses the Tabaris algorithm for finding a maximum independent set in a graph. Also compares Tabaris to another algorithm called "BALAS". Tests the algorithms on random graphs that vary in both size and density.

109. X. Geng, J. Xu, J. Xiao, L. Pan. "A Simple Simulated Annealing Algorithm for the Maximum Clique Problem." *Information Sciences*, 177: 5064-5071, 2007.

Notes: Introduces and tests a simulated annealing algorithm on all 80 DIMACS maximum clique problem instances. Tests the new algorithm against a recent efficient algorithm by Xu/Ma (ESA), and also against a trust region heuristic algorithm by Stanislav/Busygin (TR).

110. A. Grosso, M. Locatelli, W. Pullan. "Simple Ingredients Leading to Very Efficient Heuristics for the Maximum Clique Problem." *Journal of Heuristics*, 14: 587-612, 2008.

Notes: Discusses iterated local search maximum clique problem algorithms, including DLS. Introduces two versions of a new algorithm aimed at improving the currently existing algorithms – one version with a random selection rule and one with a ranking selection rule.

111. P. Hansen. N. Mladenovic, D. Urosevic. "Variable Neighborhood Search for the Maximum Clique." *Discrete Applied Mathematics*, 145: 117-125, 2004.

Notes: Introduces a new metaheuristic algorithm based on variable neighborhood search. Tests on DIMACS graphs against genetic algorithms, continuous based heuristics, tabu search, and RLS. Concludes that VNS is competitive with the current algorithms.

112. D.S. Johnson, and M.A. Trick. *Volume 26: DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

Notes: Contains several articles on the maximum clique problem by various authors.

113. K. Katayama, A. Hamamoto, H. Narihisa. "An Effective Local Search for the Maximum Clique Problem." *Information Processing Letters*, 95(5): 503-511, Sept. 2005.

Notes: Introduces the KLS algorithm, based on variable depth search (generalization of local search). Tests KLS on DIMACS benchmark graphs for the maximum clique problem on up to 4000 nodes against GENE (genetic local search), ITER (iterated local search), and RLS.

114. X. Liu, A. Sakamoto, T. Shimamoto. "A Genetic Algorithm for Maximum Independent Set Problems." *IEEE International Conference on Systems, Man, and Cybernetics*, 3: 1916-1921, Oct. 1996.

Notes: Introduces a genetic algorithm for ISP and compares it with GMCA and CBH. Tests the algorithms on the DIMACS benchmark graphs for the maximum clique problem.

115. E. Marchiori. "A Simple Heuristic Based Genetic Algorithm for the Maximum Clique Problem." *Proc. of the 1998 ACM Symp. on Applied Computing*: 366-373, 1998.

Notes: Introduces the HGA algorithm and tests it on the DIMACS benchmark graphs for the maximum clique problem against tabu search algorithms and the genetic algorithm GMCA. Concludes that HGA is competitive with the algorithms tested.

116. P.R.J. Östergård. "A Fast Algorithm for the Maximum Clique Problem." *Discrete Applied Mathematics*, 120: 197-207, 2002.

Notes: Introduces a branch-and-bound algorithm using a vertex order from a coloring as well as pruning strategies. Tests the algorithm on some of the DIMACS benchmark graphs for the maximum clique problem and also on random graphs.

117. W. Pullan, H.H. Hoos. "Dynamic Local Search for the Maximum Clique Problem." *Journal of Artificial Intelligence Research*, 25: 159-185, 2006.

Notes: Introduces DLS-MC (stochastic local search algorithm). Describes the five current best heuristic algorithms. Contains results on testing DLS-MC on all 80 DIMACS instances for the maximum clique problem. Compared DLS-MC with DAGS, GRASP, k-opt, RLS, GENE, ITER, and QUALEX-MS.

118. F. Rossi, S. Smriglio. "A Branch-and-Cut Algorithm for the Maximum Cardinality Stable Set Problem." *Operations Research Letters*, 28: 63-74, 2001.

Notes: Introduces a new branch-and-cut algorithm for the independent set problem. Tests the algorithm on DIMACS benchmark graphs for the maximum clique problem and compares it with other branch-and-bound algorithms.

119. E. Tomita, T. Kameda. "An Efficient Branch-and-Bound Algorithm for Finding a Maximum Clique with Computational Experiments." *Journal of Global Optimization*, 37: 95-111, 2007.

Notes: Introduces MCR algorithm, which uses approximate coloring and sorting of the vertices. Tests the algorithm on random graphs up to 15,000 nodes and DIMACS benchmark graphs for the maximum clique problem against dfmax, New, and COCR(COC) algorithms.

120. E. Tomita, T. Seki. "An Efficient Branch-and-Bound Algorithm for Finding a Maximum Clique." *Lecture Notes in Computer Science*, 2731: 278-289, 2003.

Notes: Introduces the algorithm MCQ, which is approved upon later by the algorithm MCR. Contains testing done on DIMACS graphs for the maximum clique problem against dfmax, New, and COCR algorithms.

121. Q. Zhang, J. Sun, E. Tsang. "An Evolutionary Algorithm with Guided Mutation for the Maximum Clique Problem." *IEEE Transactions on Evolutionary Computation*, 9(2): 192-200, April 2005.

Notes: Introduces the EA/G algorithm. Tests EA/G on the DIMACS benchmark graphs for the maximum clique problem against HGA and MIMIC. Concludes that EA/G is competitive with the other algorithms considered.

SATISFIABILITY

122. "Satisfiability Testing or How to Solve Sudoku Puzzles – The DPLL Method." From the International Center for Computational Logic. Available at: http://www.computational-logic.org/iccl/master/lectures/summer07/sat/slides/dpll.pdf

Notes: Gives a description of the satisfiability problem and an overview of the DPLL method for solving instances of the satisfiability problem.

123. G. Brassard, C. Crepeau. "Non-Transitive Transfer of Confidence: A *Perfect* Zero-Knowledge Interactive Protocol for SAT and Beyond." *Proc. of the 27th Annual Symp. on Foundations of Computer Science*: 188-195, 1986.

Notes: Outlines a basic zero-knowledge proof system for the satisfiability problem. Focuses on the commitment scheme used.

124. G. Brassard, D. Chaum, C. Crepeau. "Minimum Disclosure Proofs." *Journal of Computer and System Sciences*, 37(2): 156-189, Oct. 1988.

Notes: Describes and illustrates a zero-knowledge proof system for the satisfiability problem. Discusses how the protocol satisfies the properties necessary for a zero-knowledge protocol.

125. S. Cook, D. G. Mitchell. "Finding Hard Instances of the Satisfiability Problem: A Survey." DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 35: 1-17, 1997.

Notes: Details the algorithms DPLL, GSAT, and WalkSAT. Also does some testing and analysis of the algorithms. Has some discussion on the construction of hard satisfiability instances.

126. I. Damgard. "Non-Interactive Circuit Based Proofs and Non-Interactive Perfect Zero-Knowledge with Preprocessing." *Proc. of Eurocrypt*: 341-355, 1992.

Notes: Thoroughly outlines a noninteractive proof system for the satisfiability problem and proves that it satisfies the necessary properties. Proves that the proof system is zero-knowledge under the QRA.

127. C. Dwork, U. Feige, J. Killian, M. Naor, M. Safra. "Low communication 2-prover zero-knowledge proofs for NP." *Proc. of the 12th Annual International Cryptology Conference on Advances in Cryptology*: 215-227, 1992.

Notes: Discusses, outlines, and proves a zero-knowledge proof system for the satisfiability problem with two provers and one verifier.

128. B. Ferris, J. Froehlich. "WalkSAT as an Informed Heuristic to DPLL in SAT Solving." *Artificial Intelligence Graduate Course taught by Professor Dan Weld:* 2005. Available at: http://www.cs.washington.edu/homes/jfroehli/publications/WalkSAT-DPLL.pdf

Notes: Compares WalkSAT, a stochastic local search algorithm, with DPLL, a systematic search algorithm. WalkSAT is faster, but incomplete (cannot prove unsatisfiability), while DPLL-type algorithms are complete but slower.

129. E. Goldberg, Y. Novikov. "BerkMin: A Fast and Robust SAT-Solver." *Proc. of DATE '02*: 142-149, 2002.

Notes: Compares a new algorithm (BerkMin) with GRASP, SATO, and Chaff, which it is based off of. Tests BerkMin against these other satisfiability problem solvers and concludes that BerkMin is more robust (can solve more instances), but is not always faster.

130. H.H. Hoos, T. Stutzle. "Local Search Algorithms for SAT: An Empirical Evaluation." *Journal of Automated Reasoning*, 24(4): 421-481, 2000.

Notes: Introduces, discusses, compares and evaluates the stochastic local search algorithms WalkSAT and GSAT thoroughly.

131. H. Jia. "Hard Instances with Hidden Solutions." PhD Dissertation, University of New Mexico: December, 2007.

Notes: Introduction to several algorithms that exist for solving instances of the satisfiability problem and 3-SAT, as well as a proposed method for generating difficult test cases for these algorithms. Algorithms described: DPLL, WalkSAT, zChaff, and SP.

132. J. Marques-Silva. "The Impact of Branching Heuristics in Propositional Satisfiability Algorithms." *Proc. of the 9th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence*: 62-74, 1999.

Notes: Describes several branching heuristics that are used in effective satisfiability solvers such as GRASP, SATO, and rel_sat. Runs tests on these algorithms against other algorithms that do not use the same techniques to examine their effectiveness.

133. J.P. Marques-Silva, K.A. Sakallah. "GRASP: A Search Algorithm for Propositional Satisfiability." *IEEE Transactions on Computers*, 48(5): 506-521, May 1999.

Notes: Introduces, outlines, and discusses the GRASP algorithm for solving the satisfiability problem. Contains experimental results obtained from testing GRASP against several other well-known algorithms such as DPLL, GSAT, etc.

134. M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, S. Malik. "Chaff: Engineering an Efficient SAT Solver." *Proceedings of the 38th Conference on Design Automation*: 530-535, 2001.

Notes: Describes the Chaff algorithm for solving the satisfiability problem. Also includes a description of the DPLL algorithm as comparison, with some comments on other currently popular algorithms.

135. D.N. Pham, C. Gretton. "gNovelty$^+$." From the SAT 2007 Competition web site. Available at: http://www.satcompetition.org/2007/gNovelty+.pdf

Notes: Introduces and discusses the satisfiability solver gNovelty$^+$, based off of the first and second place winners in the random category of the 2005 SAT competition. The program was used in the 2007 SAT competition (won gold in the random SAT category).

136. H. Zhang. "SATO: An Efficient Propositional Prover." *Proc. of the 14th International Conference on Automated Deduction*: 272-275, 1997.

Notes: Describes the update to SATO 3.0 and contains some results of testing SATO 3.0 against past versions of SATO as well as other popular satisfiability solver algorithms, such as DPLL, GRASP, etc. Concludes that SATO either performs best or second best on all sets of data considered.

MINIMUM LABEL SPANNING TREE PROBLEM

137. T. Brüggemann, J. Monnot, G.J. Woeginger. "Local Search for the Minimum Label Spanning Tree Problem with Bounded Color Classes." *Operations Research Letters*, 31(3): 195-201, 2003.

Notes: Discusses the complexity of the minimum label spanning tree problem when every color appears at most *r* times in the input graph. Introduces local search algorithms for this modified problem.

138. R. Cerulli, A. Fink, M. Gentili, S. Voß. "Metaheuristics Comparison for the Minimum Labeling Spanning Tree Problem." The Next Wave in Computing, Optimization, and Decision Technologies. G. Golden, S. Raghavan, E. Wasil (Eds.), Springer-Verlag: 93-106, 2005.

Notes: Introduces new metaheuristic algorithms for the minimum label spanning tree problem. The metaheuristics implemented are SA, reactive tabu search, the Pilot method, and VNS. Compares the new algorithms with MVCA.

139. R.S. Chang, S.J. Leu. "The Minimum Labeling Spanning Trees." *Information Processing Letters*, 63(5): 277-282, 1997.

Notes: Proves that the minimum label spanning tree problem is an NP-complete problem and introduces two algorithms for approximating the solution. This paper was the first to consider this problem.

140. S. Consoli. "The Development and Application of Metaheuristics for Problems in Graph Theory: A Computational Study." Thesis for PhD in School of Information Systems, Computing and Mathematics, Brunel University, UK: November, 2008.

Notes: Introduces new algorithms for the minimum label spanning tree problem. These include GRASP, VNS, and a hybrid local search method. The new algorithms are compared to MGA (modified genetic algorithm) and the Pilot method.

141. S. Consoli, K. Draby-Downman, N. Mladenovic, J.A.M. Perez. "Greedy Randomized Adaptive Search and Variable Neighbourhood Search for the Minimum Labelling Spanning Tree Problem." *European Journal of Operational Research*, 196: 440-449, 2009.

Notes: Introduces GRASP and VNS algorithms for the minimum label spanning tree problem. Tests the algorithms against the Pilot algorithm and several others. Testing is done on graphs of order up to 500 and label sets of size up to 625 labels.

142. S.O. Krumke, H.C. Wirth. "On the Minimum Label Spanning Tree Problem." *Information Processing Letters*, 66(2): 81-85, 1998.

Notes: Proves that there cannot exist a polynomial time constant factor approximation for the minimum label spanning tree problem unless P = NP. Tests the performance of the algorithms previously created by Chang and Leu (the authors that first introduced the problem).

143. J. Nummela, B.A. Julstrom. "An Effective Genetic Algorithm for the Minimum-Label Spanning Tree Problem." *Proc. of the 8th Annual Conference on Genetic and Evolutionary Computation*: 553-558, 2006.

Notes: Introduces several new genetic algorithms for the minimum label spanning tree problem. Tests and compares the new algorithms against MVCA on random graphs.

144. Y. Xiong. "The Minimum Labeling Spanning Tree Problem and Some Variants." Thesis for PhD at the University of Maryland: 2005.

Notes: Contains an introduction to the minimum label spanning tree problem. Discusses a particularly difficult class of graphs for the MVCA algorithm. Introduces new algorithms for the problem. Tests and compares the new algorithm on random graphs.

145. Y. Xiong, B. Golden, E. Wasil. "Improved Heuristics for the Minimum Label Spanning Tree Problem." *IEEE Transactions on Evolutionary Computation*, 10(6), 700-703, 2006.

Notes: Introduces new algorithms that are either modified MVCA or modified genetic algorithms. Tests the new algorithms on random graphs and compares them to the unmodified versions of MVCA and genetic algorithms.

146. Y. Xiong, B. Golden, E. Wasil. "A One-Parameter Genetic Algorithm for the Minimum Labeling Spanning Tree Problem." *IEEE Transactions on Evolutionary Computation*, 9(1): 55-60, 2005.

Notes: Introduces a one-parameter genetic algorithm for the minimum label spanning tree problem. Tests and compares the new algorithm to MVCA. Concludes that the new algorithm is competitive with MVCA.

147. Y. Xiong, B. Golden, E. Wasil. "Worst-Case Behavior of the MVCA Heuristic for the Minimum Labeling Spanning Tree Problem." *Operations Research Letters*, 33(1): 77-80, 2005.

Notes: Analyzes the MVCA algorithm and presents a new worst-case ratio for the algorithm. Introduces a family of graphs that obtain the new ratio, proving that the ratio cannot be reduced further.

GRAPH PARTITIONING PROBLEM

148. R. Baños, C. Gil, J. Ortega, F.G. Montoya. "Multilevel Heuristic Algorithm for Graph Partitioning." *Lecture Notes in Computer Science*, 2611: 143-153, 2003.

Notes: Introduces a multilevel algorithm for solving the graph partitioning problem. Tests and compares the new algorithm with METIS, another multilevel algorithm for the problem, on the benchmark graphs maintained by Walshaw.

149. T.N. Bui, B.R. Moon. "Genetic Algorithm and Graph Partitioning." *IEEE Transactions on Computers*, 45(7): 841-855, July 1996.

Notes: Introduces hybrid genetic algorithms for the graph partitioning problem. Tests and compares the algorithms against the multistart KL algorithm and the SA algorithm on the graphs used by Johnson, et al., 1989.

150. A. Felner. "Finding Optimal Solutions to the Graph Partitioning Problem with Heuristic Search." *Annals of Mathematics and Artificial Intelligence*, 45(3-4): 293-322, Dec. 2005.

Notes: Formats the graph partitioning problem as a search problem and then applies heuristic methods to solve the problem. The algorithm does not return suboptimal solutions. Tests and compares this approach with the current best algorithms on randomly generated graphs.

151. L. Grady, E.L. Schwarts. "Isoperimetric Partitioning: A New Algorithm for Graph Partitioning." *SIAM Journal of Scientific Computing*, 27(6): 1844-1866, 2006.

Notes: Introduces a new algorithm for the graph partitioning problem based on optimization of the combinatorial isoperimetric constant. Tests and compares the algorithm against the spectral partitioning method and METIS on various classes of graphs. Concludes that the algorithm gives slightly higher averages than the other algorithms (like multilevel KL).

152. D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon. "Optimization by Simulated Annealing: an Experimental Evaluation, Part I, Graph Partitioning." *Operations Research*, 37: 865-892, 1989.

Notes: Introduces a new simulated annealing algorithm for the graph partitioning problem. Compares it to existing algorithms like KL and local optimization methods by testing the algorithms on both standard and non-standard random graphs.

153. B.W. Kernighan, S. Lin. "Partitioning Graphs." *The Bell System Technical Journal*: 291-307, Feb. 1970.

Notes: Introduces the heuristic Kernighan-Lin algorithm. Concludes that the algorithm is practical for solving large instances of the graph partitioning problem.

154. Y.H. Kim, B.R. Moon. "Lock-Gain Based Graph Partitioning." *Journal of Heuristics*, 10: 37-57, 2004.

Notes: Introduces the lock-gain based algorithm for the graph partitioning problem. Uses a new method for selecting vertices to move between partition classes. Tests the algorithm on benchmark instances from other publications (Johnson, et al., 1989, and Bui and Moon, 1996) and compares it to existing algorithms.

155. R.Z. Loureiro, A.R.S. Amaral. "An Efficient Approach for Large Scale Graph Partitioning." *Journal of Combinatorial Optimization*, 13: 289-320, 2007.

Notes: Introduces some greedy heuristic algorithms for the graph partitioning problem. Tests and compares the algorithm on benchmark instances from the graph partitioning archive maintained by Walshaw.

GRAPH DATABASES

141. The Graph Partitioning Archive. http://staffweb.cms.gre.ac.uk/~wc06/partition/ (accessed July 2010). Maintained by Chris Walshaw.

Notes: Database with test sets for the graph partitioning problem.

142. The Stanford GraphBase. http://www-cs-faculty.stanford.edu/~uno/sgb.html (accessed July 2010). Maintained by Donald Knuth.

Notes: Database with general graphs for any problem. Described in:

Knuth, Donald E. "The Stanford GraphBase: A Platform for Combinatorial Algorithms." *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1993: 41-43.

143. http://www.sergioconsoli.com/MLSTP.htm (accessed August 2009). Maintained by Sergio Consoli.

Notes: Database with test sets for the minimum label spanning tree problem.

144. The Harwell-Boeing Collection. http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/ (accessed July 2010).

Notes: Database with test sets of matrices for the minimum bandwidth problem.

145. TSPLIB. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html (accessed July 2010). Maintained by Gerhard Reinelt.

Notes: Contains instances for numerous variations of the traveling salesman problem.

146. The Graph Database. http://amalfi.dis.unina.it/graph/ (accessed July 2010). Maintained by SIVALab.

Notes: Database with test sets of graphs for the sub-graph isomorphism problem. Described in:

De Santo, M., P. Foggia, C. sansone, and M. Vento. "A Large Database of Graphs and Its Use For Benchmarking Graph Isomorphism Algorithms." *Pattern Recognition Letters* 24(2003): 1067-1079.